Messaging in the Emacs World

『Mew』

// みゅう //

1 はじめに読んでね

Mew とは、

- 電子メール
- ネットニュース
- MIME(Multipurpose Internet Mail Extensions)
- PGP(Pretty Good Privacy)

を統合し、簡単に読み書きするためのインターフェイスです。Mew を使えば、友達の誕生日にケーキの絵と「Happy Birthday to You」の歌を添えたメッセージを暗号化して送れます。ネットニュースの統合は、2.xx 以降で予定しています。

Mew は「Messaging in the Emacs World」の略です。先頭の M は大文字で表記し、「みゅう」と読みます。M で始まるかわいらしい単語を選んだ結果 Mew になりました。決して漫画の題名や某アイドルの歌、あるいは、関西の会社に因んでいるわけではありません。:p

Mew バージョン 1.9x の特長を以下に示します。

- 複雑な構造を持つメッセージを簡単に表示できます。メッセージを表示する作業は、'SPC' を押すだけです。
- コピー程度のファイルの操作を知っている人ならだれでも簡単に複雑なメッセージを作成できます。
- メッセージの一覧表示が終了するまで待たなくても、メッセージを読み始められます。
- Summary モードのメッセージの一覧を保存しているので、フォルダを移動した場合は、更新された部分だけを一覧表示します。
- メッセージの整頓先を賢く推測します (たくさんメッセージを受け取る人は、これがないと生きていけません)。
- Draft モードでは、フィールド名、アドレス、氏名、ドメイン名、フォルダ名を補完できます。
- Subject: や Date: などを条件に指定して、簡単にメッセージを選択できます。
- 便利なマークが提供されています。uuencode した後で分割されたメッセージにマークを付けて、一度に元のファイルに戻せます。
- PGP で暗号化されたメッセージを自動的に復号化します。また、電子署名を自動的に検証します。
- PGP を使って、メッセージを簡単に暗号化したり、署名したりできます。
- MIME の構造を解析したり、PGP の署名を検証したりするのには少し時間がかかります。そこで、ユーザがあるメッセージを読んでいる間に、次のメッセージをあらかじめ処理しておくことで高速性を実現しています。解析されたメッセージは、しばらくの間保存されます。
- 複数のフォルダを 1 つのフォルダに見せかけられます。
- XEmacs では、Emacs でのキー入力によるインターフェイスと全く同じ機能を持つアイコン・ベースのインターフェイスが提供されています。

Mew は、Emacs 19.28、19.34、20.3 以降、Mule 2.3 以降、および、XEmacs 20.4 以降をサポートしています。これら以外の Emacs、たとえば、Emacs 18、Nemacs、Mule 1 および XEmacs 19、20.3 などはサポートしていませんし、今後もサポートする予定はありません。

また、beta リリースである Emacs をサポートすることもありますが、正式リリースになって仕様が変わったときは、正式リリースである Emacs の仕様に合わせます。

このマニュアルで単に Emacs と言った場合には、サポートしているすべてのプラットフォームを意味します。Mule といった場合は、多国語が利用できる Mule 2 と Emacs 20、そして、–with-mule オプション付きでコンパイルした XEmacs 20.4 を指します。これに対し、Bilingual Emacs と言った場合には、英語と Latin 1 しかサポートしていない Emacs 19 と –with-mule オプション無しでコンパイルした XEmacs 20.4 を意味します。また、XEmacs と言った場合には、グラフィックを楽しめる XEmacs 20.4 を指します。反対に、テキストしか表示できない Emacs は、Text Emacs と呼びます。

2 さぁ始めよう！

Mew には次の 5 つのモードがあります。

- Summary モード :: メッセージの一覧を表示し選択するモード。
- Virtual モード :: 複数のフォルダからある条件に合致したメッセージを取り出し、仮想的に 1 つのフォルダにしたモード。Summary モードに似ている。
- Message モード :: テキストの内容を表示するモード。
- Draft モード :: メッセージの送信、返答、転送を準備するためのモード。
- Addrbook モード :: アドレス帳にエントリを登録するためのモード。

Mew を起動するには、以下の方法があります。

1. '`M-x mew`' :: Mew を起動する。'`mew-auto-get`' が '`t`' なら、到着したメッセージを +inbox フォルダに保存しながら、+inbox のメッセージを Summary モードに一覧表示。'`mew-auto-get`' が '`nil`' なら、単に +inbox のメッセージを Summary モードに一覧表示。
2. '`C-uM-x mew`' :: '`mew-auto-get`' の値を逆だと思って、'`M-x mew`' を実行する。
3. '`M-x mew-send`' :: メッセージを書くための Draft モードへ移行する。
4. '`C-xm`' :: '`mail-user-agent`' が設定されている場合、Draft モードへ移行する。

Text Emacs で Mew を起動した際には、Mew を型どった「/\\ - \\/」という図形がくるくると回るデモが始まります。XEmacs では、可愛い 2 匹の子猫がオープニングを飾ります。

到着したメッセージを +inbox フォルダに取り込む際にはパスワードを訊かれることがあります。パスワードを入力する前に以下の条件のいずれかが満たされていることを確認しましょう。

- Emacs がローカルのコンピュータで動いている
- Emacs がリモートのコンピュータで動いているが、なんらかの暗号手段を使って通信している。

どちらの条件も満たされない場合は、パスワードを入力しないで下さい。入力すると盗聴される恐れがあります。

もし Mew が起動されない場合は、Mew や IM がインストールされているか、また以下が組織の設定ファイルか自分の .emacs で設定されているか確かめて下さい。

```
(autoload 'mew "mew" nil t)
(autoload 'mew-send "mew" nil t)
(setq mew-mail-domain-list '("your mail domain"))
(setq mew-icon-directory "a directory where Mew's image files are installed.")
(autoload 'mew-user-agent-compose "mew" nil t)
(if (boundp 'mail-user-agent)
    (setq mail-user-agent 'mew-user-agent))
(if (fboundp 'define-mail-user-agent)
    (define-mail-user-agent
      'mew-user-agent
      'mew-user-agent-compose
      'mew-draft-send-letter
      'mew-draft-kill
      'mew-send-hook))
```

3 メッセージを表示する

‘M-x mew’ と入力すると、Mew はスプールのメッセージを +inbox フォルダに移して以下のように一覧表示します。

```
  1  07/17 いとぢゅん     v6: items to be no in6_pcbnotify() がなにも
  2  07/18 歌代先生       Re: behavior after これ、mark-ring がどんど
  3  07/19 のむさん       refile info.       乃村です。遅くなりました。
```

これを Summary モードといいます。ここでは、主に Summary モードでのメッセージの読み方について説明します。

3.1 読み方の基礎

メッセージを上から順に読んでいくのであれば、‘SPC’ を適宜押すだけです。簡単でしょ？

しかしそれだけではあんまりなので、以下にページを操作する基本的なコマンドを示します。

‘SPC’ メッセージを読み進める。つまり、メッセージを表示し、スクロールさせ、他のメッセージに移動して表示する。カーソルが移動する方向は、See Section 9.1 [level-one], page 43 を参照のこと。

‘DEL’ 現在のメッセージを上にスクロールさせる。不必要なヘッダフィールドは、ウインドウの上に隠れている。よって、‘DEL’ を入力すると、それらが現れる。

‘.’ メッセージが ‘mew-file-max-size’ を越えている場合は、MIME の解析が中止され、メッセージの先頭部分が生でメッセージ・バッファに表示される。このような場合に、‘.’ を押すと、強制的に MIME が解析されメッセージが表示される。

‘,’ MIME 解析をしていない生のメッセージを表示する。メッセージの内 ‘mew-file-max-size’ で指定されたバイト数以下の部分を表示する。‘C-u’ と共に呼ばれると、メッセージ全体を生で表示する。

‘RET’ 現在のメッセージを 1 行上にスクロールする。

‘M-RET’

‘-’ 現在のメッセージを 1 行下にスクロールする。

‘C-n’ 下の行へ移動。

‘C-p’ 上の行へ移動。

‘n’ 下方向に移動し表示。対象となるのは、パート、‘*’ マークの付いたメッセージ、マークの付いていないメッセージ。‘C-u’ と共に呼ばれると、パート部分を飛ばす。

‘p’ 上方向に移動し表示。対象となるのは、パート、‘*’ マークの付いたメッセージ、マークの付いていないメッセージ。‘C-u’ と共に呼ばれると、パート部分を飛ばす。

‘j’ 入力された番号にしたがってメッセージへ移動。

3.2 MIME を表示する

マルチパートを読むのは別に大変なことではありません。今まで通り、'SPC' を押していけばよいだけです。

マルチパートのメッセージは、以下のように日付の左に "M" という印が付いています。

```
      4  07/19 しげやさん     Re: imget very fir ということで、こんなもん
      5 M07/20 いとぢゅん     MagicPoint         今度の発表で使用する資料
      6  07/21 もとのりさん   Re: imget very fir POP でサーバにメッセージの
```

"5" のところで、'SPC' を押すと、ヘッダを Message モードに表示すると共に、以下のように Summary モードでマルチパートの構造を簡素に表示します。

```
      4  07/19 しげやさん     Re: imget very fir ということで、こんなもん
      5 M07/20 いとぢゅん     MagicPoint         今度の発表で使用する資料
    B    2  Image/Gif                    MagicPoint のロゴ          mgp.gif
    Q    3  Application/Postscript       資料                       ohp.ps
      6  07/21 もとのりさん   Re: imget very fir POP でサーバにメッセージの
```

もし、パート 1 が Text/Plain なら、Summary モードにはパート 1 は可視化されず、そのかわりパート 1 がヘッダと共に Message モードに表示されます。

マルチパートの各行は

- マーク (Content-Transfer-Encoding:)
- パート番号
- データの型 (Content-Type:)
- 説明 (Content-Description:)
- ファイル名 (Content-Disposition:)

から構成されています。Content-Description: はパートに対する Subject: と考えていいでしょう。この表示は Draft モードの添付領域とほとんど同じです。それぞれのカラムの詳しい意味は、See Section 4.6 [mime-comp], page 19 を参照して下さい。

'SPC' や 'n' でパート 1 へ進めば、そのパートがデータ型に応じて表示されます。たとえば、Text/Plain なら Message モードで、PostScript なら ghostview で表示されます。

'n' や 'p' は、パート部分まで含んだ行を移動することに注意して下さい。パート部分を飛ばして下のメッセージを表示するには 'C-u n' と入力して下さい。また、パート部分を飛ばして 1 つ上のメッセージを表示するには 'C-u p' と入力して下さい。

Mew は、MIME を再帰的に処理します。以下は転送されたマルチパートのメッセージの例です。

```
    501 M02/22 Itojun         Fw: MagicPoint          萩野先生からこんな
      2  Message/Rfc822              MagicPoint
    B   2.2  Image/Gif                MagicPoint のロゴ          mgp.gif
    Q   2.3  Application/Postscript   資料                       ohp.ps
```

(メモ) テキスト以外のデータを、シングルパートとしてメッセージに格納するのは、書式としては間違いではありませんが、お勧めできません。マルチパートを作成し、そのパート 1 に説明のテキストを、パート 2 にテキスト以外のデータを入れる作法をお勧めします。

テキスト以外のデータが本文に直接格納されているメッセージに対し、Mew はこれをあたかもマルチパートのように表示します。

このように MIME の構造は複雑になりうるので、解析するのには時間がかかる場合があります。しかし、Mew は次に読まれるメッセージを予想し、ユーザが現在のメッセージを読

んでいる間に、次のメッセージをあらかじめ解析しておくことで高速性を実現しています。解析されたメッセージは、しばらくの間保存されます。

メッセージの終りの部分がはっきりと分かるように、Mew はメッセージの最後に"[End of message]" という文字列を表示します。また、パートの終りでは、"[Message is continued]" という文字列を表示します。(Emacs の機能不足により色は付きません。) この機能は Emacs 19.28 やそれを基にしている Mule 2.3 などでは利用できません。これらの文字列はそれぞれ、'`mew-end-of-message-string`' と '`mew-end-of-part-string`'で指定できます。

3.3 PGP/MIME を表示する

今までと同様 '`SPC`' などを利用することで、Mew では PGP で暗号化や電子署名を施されたメッセージを簡単に表示できます。まず、簡単な例から紹介しましょう。

```
  8 S07/22 酒井さん        Re: home was full  MsgStore.pm のバグです
  9 E07/23 ニートすみかわ Wine                おはようからおやすみま
```

8 番と 9 番のメッセージには、それぞれ "S" マークと "E" マークが付いています。これはそれぞれ、本文全体が署名されている、および、暗号化されていることを意味します。

PGP/MIME では、一部のパートに電子署名を施したり、暗号化したりできます。この場合このようなマークは付きません。マークが付くのは、本文全体が対象になっている場合です。

また、単に署名や暗号化と言いましたが、これは最終的な処理が署名や暗号化であったことを意味しています。やや複雑な話になりますが、もしかすると前者は本文全体を暗号化した後署名したのかもしれませんし、後者は一部のパートに署名しさらに全体を暗号化している可能性もあります。

本文全体、あるいは、一部のパートが暗号化されている場合、Mew はパスフレーズを訊いてきます。入力されたパスフレーズは、あなたの秘密鍵を復号化するのに使われます。そして、復号化された秘密鍵によって、暗号化されているメッセージを解くわけです。

ある PGP/MIME メッセージを表示するには、暗号化された数だけパスフレーズを入力する必要があります。これは Mew が安全を期して、通常パスフレーズを保存しないからです。もしこれがわずらわしいなら、以下の設定で Mew にパスフレーズをしばらくの間保存させることも可能です。パスフレーズを保存する際の注意事項については、See Section 3.4 [folder], page 7 を参照して下さい。

```
(setq mew-use-pgp-cached-passphrase t)
```

パスフレーズを保存しない通常の設定でも、一旦復号化されたメッセージはしばらく保存されるので、2 回目の表示にはパスフレーズを訊かれないかもしれません。

一方、通信相手の署名を検証するためには相手の公開鍵が利用できればよいので、パスフレーズを訊かれることはありません。

Mew は自動的に電子署名を検証したり、入力されたパスフレーズを使って暗号メッセージを復号化したりして、元のメッセージを表示します。そこで、ユーザが署名の存在に気づかないかもしれませんし、どの部分が暗号化されていたのか分からないかもしれません。

そこで、検証の結果やどの部分が暗号化されていかをユーザに通知するために、Mew は以下のようにヘッダに X-Mew: フィールドを挿入します。

```
X-Mew: <body> PGP decrypted.
       Good PGP sign "kazu@mew.org" COMPLETE
```

"<>" 内の番号は、どのパートの結果であるかを示しています。"body" は、メッセージの本文全体が保護されていることを意味します。この例では、メッセージ全体が kazu によって

署名され、受信者のために暗号化されています。Mew はまずこれを復号化し、そして署名を検証しています。署名は正しいので、kazu@mew.org という ID の付いた秘密鍵で署名されたときから、内容は変更されていないと分かります。この署名の検証に使われた公開鍵の「有効性」は"complete" です。よって、受信者はこの公開鍵が ID が示すユーザに属していると信頼していることになります。つまり、このメッセージは信頼をおいている公開鍵によって検証され結果が正しいので、改竄されていないということになります。

以下の例では、まずマルチパートである本文全体の電子署名が検証され、その後パート 2 のメッセージ全体が復号化されています。つまり、作成時には、まずパート 2 が暗号化され、そして本文全体が署名されたことが分かります。

```
X-Mew: <body multi> Good PGP sign "kazu@mew.org" COMPLETE
X-Mew: <2 message> PGP decrypted.
```

するどい人なら、悪い人があらかじめ X-Mew: フィールドを挿入しておき、あなたをだますかもしれないと思うかもしれません。でも大丈夫です。Mew は、ヘッダにある X-Mew: をあらかじめ削り、本物の X-Mew: フィールドを挿入しますから。

X-Mew: フィールドは他にもさまざまな結果を伝えてくれます。たとえば、署名を検証するための公開鍵がないとか、復号化に失敗したなどです。以下の例は、鍵 ID が 0x1B8BF431 である公開鍵がないことを示しています。

```
X-Mew: <body multi> No his/her public key. ID = 0x1B8BF431
```

この場合、‘C-cC-f’ と入力すると、Mew は‘mew-pgp-keyserver-url-template’ で指定された URL を使ってこの公開鍵の入手を試みます。もし、X-Mew: フィールドがない場合は、‘C-cC-f’ は From: を ID と考えます。また、‘C-uC-cC-f’ は、X-Mew: フィールドに加えて ‘mew-x-pgp-key-list’ に指定されたフィールドも鍵 ID を切り出す対象とし、公開鍵の入手を試みます。

Mew では PGPv2、PGPv5、GNUPG に対応しています。これらは Summary モードにおいて、‘C-cC-v’ で切替え可能です。これら複数の PGP を使いたい人は‘mew-prog-pgp2’、‘mew-prog-pgp5’、‘mew-prog-gpg’ に対し、それぞれ PGPv2、PGPv5、GNUPG のコマンド名を設定して下さい。また、Mew の起動直後に利用する PGP のコマンド名を‘mew-prog-pgp’ に設定して下さい。なおパスフレーズは、それぞれの PGP に対し独立に保存されます。

3.4 フォルダの更新と移動

到着したメッセージを +inbox フォルダに移動し、一覧を表示するには‘i’ を使います。一覧は、+inbox フォルダの Summary モードの最後に挿入されます。

このときパスワードを訊かれることがあります。パスワードを入力する際の注意事項については、See Chapter 2 [Start], page 3 を参照して下さい。パスワードを何回も入力するのが面倒な人は、パスワードを保存する機能を利用して下さい。これには以下設定が必要です。

```
(setq mew-use-cached-passwd t)
```

パスワードの保存機能を使うと、パスワードが保存されている間は、パスワードの入力を省略できます。パスワードは Emacs の中に保存されていますので、席を空けて Emacs が他の人に使用されることがないよう注意して下さい。

フォルダの一覧を再表示するには、‘s’ を使います。このコマンドを対話的に使うと範囲を訊いてきます。Mew で重要な範囲の意味を以下に示します。

‘update’ 「Summary モードの最後のメッセージの次」から「フォルダ内の最後のメッセージ」まで。つまり、Summary モードと実際のフォルダ内のメッセージの差分。

‘all’ フォルダ内のメッセージすべて。Summary モードの表示がおかしくなったときに、内容を一新するために用いる。

+draft 以外のフォルダでは、デフォルトの範囲が ‘update’ となっています。ですから、‘s’ の後に ‘RET’ を押すだけで、現在のフォルダに対し最新の一覧を得られることになります。+draft のデフォルトの範囲は ‘all’ です。

Mew ではあまり重要ではありませんが、以下の範囲も指定できます。

‘<num1>-<num2>’
<num1> から <num2> まで。

‘<num>:N’ <num> から N 個。

‘<num>:-N’
<num> までの N 個。

‘first:N’ 最初から N 個。

‘prev:N’ 現在のメッセージまでの N 個。

‘next:N’ 現在のメッセージから N 個。

‘last:N’ 最後のメッセージまでの N 個。

フォルダの移動には ‘g’ を入力して下さい。フォルダ名は ‘TAB’ で補完できます。もし、移動した際に Summary モードの一覧が古いと判断した場合は、自動的に差分を追加表示します。

出てきたコマンドを以下にまとめます。

‘i’ 到着したメッセージを +inbox フォルダに移動し一覧を表示する。

‘s’ フォルダの一覧を再表示する。

‘g’ フォルダを移動する。

3.5 送信、返答、転送

メッセージの送信、返答、転送には、以下のコマンドを使います。

‘w’ メッセージを書く。新しい草稿が Draft モードに用意される。

‘a’ 現在のメッセージに返答する。Draft モードに草稿が用意され、To: や Cc: が自動的に決定される。

‘A’ 現在のメッセージに返答する。Draft モードに草稿が用意され、To: や Cc: が自動的に決定された後、本文が引用される。

‘f’ 現在のメッセージを第 3 者に転送する。Draft モードに草稿が用意され、現在のメッセージが自動的に添付される。

‘F’ ‘@’ マークの付いたメッセージを第 3 者に転送する。Draft モードに草稿が用意され、‘@’ マークの付いたメッセージが自動的に添付される。詳しくは See Section 5.3 [multi mark], page 31 を参照のこと。

エラーメッセージが返ってきたら、以下のコマンドで修正し再挑戦しましょう。

‘E’ メッセージの再編集。または、MIME 形式でカプセル化されて戻ってきたメッセージの再編集。+draft フォルダでは、そのメッセージを直接編集。その他のフォルダでは +draft フォルダにコピーしてから編集。

‘M-e’ "—— Original message follows ——" の後にオリジナルのメッセージが引用されているエラーメッセージの再編集。

3.6 便利な機能

Mew では、Summary モードに以下のような便利なコマンドが用意されています。

‘v’ 「Summary モードのみ」と「Summary & Message モード」の切り替え。「Summary モードのみ」を選んでいる場合は、‘d’ は次のメッセージを表示しないので、連続してすばやく ‘D’ マークを付けられる。

‘M-l’ 現在の行を Summary モードの中央に移動させる。

‘C-cC-s’ Message モードで順方向段階的検索。

‘C-cC-r’ Message モードで逆方向段階的検索。

‘y’ メッセージ、あるいは、パートを入力したファイル名で保存する。Mule 上で ‘C-u’ と共に呼び出すと、保存するテキストの coding-system を指定できる。

‘#’ 現在のメッセージかパートを印刷する。

‘|’ 現在のメッセージかパートをパイプで出力する。

‘O’ メッセージを番号詰めして、再び一覧表示する。

‘B’ 格納されているメッセージを取り出す。

‘D’ +trash フォルダのメッセージを全部消去する (See Section 5.1 [delete mark], page 30)。

‘Z’ アドレス帳 (See Section 4.3 [addrbook], page 15) を読み込んで情報を更新する。‘C-uZ’ では、アドレス帳に加えフォルダのリストも更新する。‘mew-use-folders-file-p’ が ‘t’ ならフォルダのリストを"~/Mail/.folders" に保存する。デフォルトは ‘t’。

‘C-cC-a’ 現在のメッセージの情報をアドレス帳に登録する (See Section 4.3 [addrbook], page 15)。

‘C-cC-v’ PGP のバージョンを切替える (See Section 3.3 [pgp-viewing], page 6)。

‘C-cC-p’ 昔ながらの自動処理できない PGP メッセージを PGP に復号化、検証させる。

3.7 メッセージのソート

フォルダ内のメッセージをソートするには ‘S’ を使います。このとき次のようにどのフィールドでソートするか訊いてくるので、ソートしたいフィールド名を入力して下さい。

```
Sort by? (default date):
```

指定したフィールドに書かれている文字列は単純に文字列比較でソートするべきではありません。たとえば、Subject: はテキストと考えてよいのですが、Date: は日付、X-Mail-Count: は数字と考えてソートすべきです。このように文字列をどう取り扱うかをモードと言います。

ソートによく指定されるフィールド名に対するデフォルトのモードは ‘mew-sort-key-alist’ で設定されています。

ソートのモードを明示的に指定、変更したい場合には ‘:’ で区切って指定します。たとえば X-Mail-Count フィールドの内容を (テキストとしてではなく) 数値とみなしてソートしたい場合には、次のように入力します。

```
x-mail-count:num
```

なお、文字の大文字、小文字は区別しません。また、フィールド名やモードは ‘TAB’ で補完できます。

"Sort by?" と訊かれる際のデフォルトのフィールド名は、‘mew-sort-default-key’ で設定できます。以下は、デフォルトの "date" を "x-ml-count" に変更する例です。

```
(setq mew-sort-default-key "x-ml-count")
```

‘mew-sort-default-key-alist’ で、フォルダごとにデフォルトのフィールド名を設定することもできます。ここで指定しなかったフォルダでは、デフォルトのフィールド名として ‘mew-sort-default-key’ の値が使われます。以下は、+inbox フォルダでのデフォルトを "subject" に、+mew-dist フォルダでのデフォルトを "x-mail-count" に変更する例です。

```
(setq mew-sort-default-key-alist
      '(("+inbox" . "subject")
        ("+mew-dist" . "x-mail-count")))
```

ソートに関するコマンドをまとめると以下のようになります。

‘S’ 入力したフィールドを用いてフォルダ内のメッセージをソートします。‘C-u S’ のように prefix を付けると、リージョン内のメッセージのみをソートします。

‘mS’ ‘*’ マークの付いたメッセージをソートします。

3.8 化けたメッセージ

以下のようなメッセージは、charset がなく、US-ASCII と認識されてしまうので、化けます。

```
To: kazu
Subject: 化けるメッセージ
From: Alice
MIME-Version: 1.0
Content-Type: Text/Plain

日本語の本文
```

このような場合は、‘C-cC-l’ を押すと、文字コードを正しく変換し表示します。

また以下のようにメッセージのヘッダが化けることがあります。

```
From: "=?iso-2022-jp?B?GyRCOzNLXE9CSScbKEI=?=" <kazu@iijlab.net>
```

上の例では "=?" と "?=" で囲まれた部分はもともと日本語でした。メッセージの規格ではヘッダには ASCII のみが格納できると定められています。よって、ASCII 以外の文字コードをヘッダに格納するには、ある規則に従って ASCII に符号化する必要があります。しかし、この符号化された文字列を 「"」 で囲むのは間違いです。「"」で囲まれた文字列は、そのままの形で取り扱われます。よって、上の例の "=?" と "?=" で囲まれた部分が日本語に復号化されることはありません。

規格に無頓着な一部のメーラではこのような間違いを平気で犯します。正しい対処方法は、このようなメーラの作者に頼んで、規格を正しく実装するように変更してもらうことです。しかしそれまで待てない人は、以下の設定をして下さい。これで「"」中の "=?" と "?=" で囲まれた部分が、Summary モードでも Message モードでも復号化されます。

```
(setq mew-decode-quoted t)
```

4 メッセージを作成する

ここではメッセージの作成方法について説明します。Mew では、MIME 形式のメッセージだけを作成できます (MIME-Version: のないメッセージは作成できません)。

新しいメッセージを書くために、Draft モードに移行するには、次の手段があります。

1. 'M-x mew-send' と入力する。
2. 'mail-user-agent' が設定されている場合、'C-xm' と入力する。
3. Summary モードで 'w' を押す。

すると、以下のようなバッファが用意されます。

```
To:
Subject:
X-Mailer:Mew version 1.94 on XEmacs 20.4
----
```

これを Draft モードといいます。Draft モードにおいて、"—-" より上をヘッダ、下を本文と呼びます。

またメッセージへの返答 ('a' や 'A') や転送 ('f' や 'F') でも Summary モードから Draft モードへ移行します。

草稿は、+draft フォルダの下に作成されます。同時に複数の草稿を持つことが可能です。

以下、Draft モードの使い方を説明します。

4.1 ヘッダの補完

ヘッダでは 'TAB' に対し、以下ように各フィールド用の完機能が割り当てられています。

– フィールド名の補完
– アドレスの短縮名の補完と展開 (To:、Cc: など)
– フォルダ名の補完 (Fcc:)

<フィールド名の補完>

行頭の単語中で、しかも、上の行の最後が ":" で終る継続行でなければ、'TAB' で 'mew-fields' に定義されているフィールド名を補完できます。

```
To: kazu@mew.org
R'TAB'
```

上記の場所で 'TAB' を押すと以下のようになります。

```
To: kazu@mew.org
Reply-To:
```

<アドレスの短縮名の補完と展開>

Mew では、アドレス帳という機能を使って、長く分かりにくいアドレスに短く覚えやすい短縮名を付けられます。たとえば、以下のように設定したとします。

```
pooh:           winnie-the-pooh@100acre.woodwest.uk
```

これは、アドレス "winnie-the-pooh@100acre.woodwest.uk" に "pooh" という短縮名付けていることになります。この短縮名は、通常 "~/.im/Addrbook" というファイルに設定します。アドレス帳の機能の詳細については、See Section 4.3 [addrbook], page 15 を参照して下さい。

Draft モードのヘッダ内で、かつ、アドレスを書くべきフィールド上で、しかも、1 文字以上の文字列が前にある場所で ‘TAB’ を打つと、アドレスの短縮名が補完されます。

例を挙げてみます。

```
To: piglet@beech.tree.uk,
        po‘TAB’
```

このように ‘TAB’ を押すと、(他に候補が無ければ) "pooh" まで補完されます。

```
To: piglet@beech.tree.uk,
        pooh‘TAB’
```

もう一度 ‘TAB’ を押すと "winnie-the-pooh@100acre.woodwest.uk" に展開されます。

```
To: piglet@beech.tree.uk,
        winnie-the-pooh@100acre.woodwest.uk
```

アドレスが補完できない場所で ‘TAB’ を打つと、単に ‘TAB’ が入ります。たとえば、以下の例を考えて下さい。

```
To: pooh,‘TAB’
```

この場合、単に ‘TAB’ が挿入されます。

"@"で終る文字列は強制的に展開します。たとえば、以下のように似たような短縮名があった場合を考えて下さい。

```
pooh:           winnie-the-pooh@100acre.woodwest.uk
pooh-pooh:      pooh-pooh@somewhere.jp
```

"pooh" を "winnie-the-pooh@100acre.woodwest.uk" に強制的に展開するには、以下のようにします。

```
To: pooh@‘TAB’
```

<フォルダ名の補完>

Fcc: などのようにフォルダを補完すべきところでは、‘TAB’ でフォルダを補完できます。以下例を挙げてみます。

```
Fcc: ‘TAB’
```

"+" が補完されます。

```
Fcc: +‘TAB’
```

‘TAB’をもう 1 度押すと候補が表示されるので、候補を見ながら適切な文字を入力します。

```
Fcc: +B‘TAB’
```

候補が一意に定まれば補完されます。

```
Fcc: +Backup
```

<設定のヒント>

アドレスの短縮名とフォルダ名をどのフィールドで補完できるようにするかは、‘mew-field-completion-switch’ で定義できます。デフォルトでは以下のように宣言されています。

```
(defvar mew-field-completion-switch
  '(("To:"        . mew-complete-address)
    ("Cc:"        . mew-complete-address)
    ("Dcc:"       . mew-complete-address)
    ("Bcc:"       . mew-complete-address)
    ("Reply-To:"  . mew-complete-address)
```

```
        ("Fcc:"       . mew-complete-folder)
        ("Resent-To:" . mew-complete-address)
        ("Resent-Cc:" . mew-complete-address)
        ("Config:"    . mew-complete-config)))
```

Config: の補完に関しては See Section 9.5 [config], page 46 を参照して下さい。

4.2 ヘッダの循環的な補完

ヘッダでは、‘C-cTAB’ に循環的な補完機能が割り当てられています。循環的な補完機能とは、あるリストのある値がそのリストの次の値に置き換えられることです。リストの最後は、最初につながっていると考えます。ヘッダ中の循環的な補完機能は、以下のようにフィールドごとに異なります。

– ドメイン名の循環的な補完 (To:、Cc: など)

– From: の循環的な補完 (From:)

<ドメイン名の循環的な補完>

アドレスを書くべきフィールドでは、‘C-cTAB’ でドメインを補完します。補完の候補は ‘mew-mail-domain-list’ から選ばれます。

```
To: kazu@‘C-cTAB’
```

上記の場所のように候補が一意に定まらない場合は、‘mew-mail-domain-list’ の最初のドメイン名が挿入されます。

```
To: kazu@mew.org‘C-cTAB’
```

補完された後、さらに ‘C-cTAB’ を押すと ‘mew-mail-domain-list’ の次の候補に変換します。

```
To: kazu@wide.ad.jp
```

また、以下の補完が一意に定まれば、その候補を挿入します。

```
To: kazu@w‘C-cTAB’
```

上記の例は次のようになります。

```
To: kazu@wide.ad.jp
```

<From: の循環的な補完>

From: フィールド上では、‘C-cTAB’ は ‘mew-from-list’ の値を循環的に補完します。このリストの最初の値 (別名 ‘mew-from’) は、次のように既に挿入されているかもしれません。

```
From: Kazu Yamamoto (山本和彦) <Kazu@Mew.org>
```

値の場所ならどこでも構いませんが、‘C-cTAB’ と入力すると、この値を ‘mew-from-list’ の次の値と置き換えます。たとえば、

```
From: Kazu Yamamoto (山本和彦) <Kazu@Mew.org>‘C-cTAB’
```

は以下のようになります。

```
From: Kazuhiko Yamamoto <kazu@wide.ad.jp>
```

循環的な補完のフィールドと関数の対応は、‘mew-field-circular-completion-switch’ で定義できます。デフォルトでは以下のように宣言されています。

```
(defvar mew-field-circular-completion-switch
  '(("To:"          . mew-circular-complete-domain)
    ("Cc:"          . mew-circular-complete-domain)
```

```
("Dcc:"         . mew-circular-complete-domain)
("Bcc:"         . mew-circular-complete-domain)
("Reply-To:"    . mew-circular-complete-domain)
("Resent-To:"   . mew-circular-complete-domain)
("Resent-Cc:"   . mew-circular-complete-domain)
("From:"        . mew-circular-complete-from)
("Resent-From:" . mew-circular-complete-from)
("Config:"      . mew-circular-complete-config)))
```

Config: の循環的な補完に関しては See Section 9.5 [config], page 46 を参照して下さい。

4.3 アドレス帳

Mew 1.94 からアドレスの alias とペットネームがアドレス帳に統合されました。alias ("~/.im/Aliases") とペットネーム ("~/.im/Petname") は今後積極的には保守されませんので、できればアドレス帳 ("~/.im/Addrbook") に乗り換えて下さい。アドレス帳には 2 つの書式が用意されています。一方は「展開規則」を指定する書式、他方は「個人情報」を記述するための書式です。

まず、「展開規則」を記述するための書式を示します。

```
<shortname>: <address1>[, <address2>, <address3>,...]
```

このように短縮名と展開すべきアドレスを ':' で区切って書きます。複数のアドレスに展開する場合は、それらのアドレスを ',' で区切ります。(これは、To: などでアドレスが ',' で区切られているのと同じです。) ',' の後ろに空白を入れても構いません。以下に例を示します。

```
pooh:           winnie-the-pooh@100acre.woodwest.uk
piglet:         piglet@beech.tree.uk
friends:        pooh, piglet
```

Mew では多段の展開が可能です。たとえばこの例で、次のように "friends" を展開してみましょう。

```
To: friends‘TAB’
```

"friends" 内部で "pooh" と "piglet" に展開され、さらにそれぞれが展開されるので、次のようになります。

```
To: winnie-the-pooh@100acre.woodwest.uk, piglet@beech.tree.uk
```

次に、「個人情報」を記述するための書式を示します。

```
<shortname> <address1>[, <address2>, <address3>,...] <nickname> <fullname>
```

このように 4 つの要素を空白で区切ります。<shortname> が短縮名です。<nickname> と <fullname> はそれぞれニックネームと正式な氏名であり、日本語でも構いません。2 番目の要素はアドレスです。複数のアドレスをその人が持っている場合は、',' で区切って書きます。',' の後に空白を入れても構いません。つまり、この空白は要素の区切りではありません。また、'"' で囲まれた空白も要素の区切りにはなりません。以下に例を示します。

```
kazu kazu@mew.org, kazu@iijlab.net Kazu-kun  "Kazuhiko Yamamoto"
```

「展開規則」の書式の場合と違って、「個人情報」の書式では、アドレスが順に置き換えられていきます。以下の例を考えて下さい。

```
To: kazu‘TAB’
```

"kazu" の後で 'TAB' を打つと、"kazu@mew.org" に置き換わります。

```
To: kazu@mew.org‘TAB’
```

次に "kazu@mew.org" の後で ‘TAB’ を打つと、"kazu@iijlab.net" に置き換わります。

```
To: kazu@iijlab.net‘TAB’
```

さらに "kazu@iijlab.net" の後で ‘TAB’ を打つと、"kazu@mew.org" に戻ります。このように ‘TAB’ を押すと、アドレスが循環的に置換されます。アドレスを決定した後は、正式名称が付加できます。

```
To: kazu@mew.org‘M-TAB’
```

このように ‘M-TAB’ を押すと、以下のように正式名称が付加されます。

```
To: Kazuhiko Yamamoto <kazu@mew.org>
```

「個人情報」の書式では、各要素を省略できます。中間の要素を省略する場合は、‘*’ と書いて下さい。以下に、アドレスに対してニックネームのみを定義する例を示します。

```
* kazu@mew.org, kazu@iijlab.net Kazu-kun
```

ニックネームは Summary モードでのアドレスの置き換えと、Draft モードでの引用記号の置き換え (See Section 4.5 [cite], page 18) に利用されます。

アドレス帳のコメント文字は ‘;’ と ‘#’ です。‘;’ は行頭にある場合のみコメントとなり、その行が無視されます。‘#’ は任意の場所でコメントとなり、そこから行末までが無視されます。

実はアドレス帳以外にも、自動的に追加される短縮名があります。メッセージを送信した場合、To: と Cc: にあるアドレスは、ユーザ名が短縮名として登録されます。以下の例を考えて下さい。

```
To: kazu@mew.org
```

このメッセージを送信すると、アドレス "kazu@mew.org" に対し、短縮名"kazu" が自動登録されます。ただし、すでに "kazu" という短縮名が自動登録されているなら、‘mew-addrbook-override-by-newone’ の値に応じて上書きするかを決定します。‘nil’ なら古い設定を残し、それ以外なら上書きします。展開の際は、アドレス帳の方が優先されます。アドレス帳に無い短縮名のみが有効になります。自動登録されるのは通常 1000 個 (‘mew-lisp-max-length’) のアドレスまでです。それを越えて登録すると古いものから消えていきます。これらの情報は Mew を終了する際に、"~/Mail/.mew-alias" に保存されます。

Summary モードには、現在読んでいるメッセージの情報を Addrbook に登録する機能があります。展開規則を登録するには ‘C-cC-a’、個人情報を登録するには ‘C-uC-cC-a’ と入力して下さい。以下に個人情報を登録している例を示します。

```
#If you want to register this entry, type C-c C-c.
#If you want to NOT register this entry, type C-c C-q.
Shortname: kazu
Addresses: kazu@mew.org
Nickname:
Name: Kazuhiko Yamamoto
Comments:
```

必要であれば加筆訂正します。実際に登録するには ‘C-cC-c’、登録を取り止める場合は ‘C-cC-q’と入力して下さい。

4.4 メッセージの送信

草稿を書き上げ送信する準備ができたら、'C-cC-m C-cC-c' と入力して下さい。

たとえば、以下のようなメッセージを送る場合を考えます。

```
To: pooh
Subject: PGP/MIME を使おうよ
X-Mailer:Mew version 1.94 on XEmacs 20.4
----
Mew がセキュリティ・マルチパートをサポートしました。

--かず
```

'C-cC-m' まで入力すると、以下のようになります。

```
To: winnie-the-pooh@100acre.woodwest.uk
Subject: PGP/MIME =?iso-2022-jp?B?GyRCJHI7SCQqJCYkaBsoQg==?=
X-Mailer:Mew version 1.94 on XEmacs 20.4
Mime-Version: 1.0
Content-Type: Text/Plain; charset=iso-2022-jp
Content-Transfer-Encoding: 7bit

Mew がセキュリティ・マルチパートをサポートしました。

--かず
```

ここで注意して頂きたいのは、Content-Type: に Text/Plain を選び、charset を推測していることです。

次に 'C-cC-c' と入力すると通常のテキストメッセージを送れます。メッセージはバックグランドで送信されます。このように 'C-cC-m' は MIME の作成、'C-cC-c' は送信です。Mew では「できるだけ見たままのメッセージを送信する」というポリシーがあるので、ユーザに明示的に MIME を作ってもらうことにしています (今までに余分な .signature や Fcc: が勝手に付けられていやな思いをした人はいませんか？ :p)。

エラーが起きた場合は、"*Mew watch*" バッファを表示します。エラーが生じて消えなかった "*Mew watch*" バッファは、'C-cC-q' で消せます。多くの場合、草稿は +draft フォルダに残っています。そこで、Summary モードで 'g' と押して +draft フォルダに移動し、+draft フォルダの Summary モードで 'E' を押して再編集して下さい。

メッセージの送信が終っていないのに、'C-xC-c' で Emacs を終了させようとすると、

```
Active processes exist; kill them and exit anyway? (yes or no)
```

と訊かれます。"*Mew watch*" バッファがなくなってから終了させて下さい。

'C-cC-m' は省略できます。この場合、MIME を自動的に作った後に

```
The header was modified. Send this message? (y or n)
```

と訊いてきますので、'y' を入力して下さい。Mew では、見たままのメッセージを送信することをモットーにしているので、Mew が勝手にメッセージを書き換えた場合は、このように質問を受けます。

すべてのメッセージ作成方法で 'C-cC-m' を省略して構いません。ただし、見たままのメッセージを送信したい人は、'C-cC-m C-cC-c' と入力する癖を付けることをお勧めします。

カーソルのある場所に "~/.signature" を挿入するコマンドは 'C-cTAB' です。シグニチャファイルは、'mew-signature-file' で設定できます。'mew-signature-as-lastpart' や 'mew-signature-insert-last' を設定することで、'C-cTAB' の動作をカスタマイズできます。

出てきたコマンドを以下にまとめます。

'C-cC-m' MIME を作成する。Charset の推測、ファイル構造をマルチパートへ変換など。

'C-cC-c' メッセージを送信する。

'C-uC-cC-c'
メッセージを送信するが、草稿は消さない。複数の人に内容を少しずつ変えながらメッセージを送りたい場合に便利。

'C-cTAB' カーソルの位置に "~/.signature" を挿入する。

4.5 引用

Summary モードの 'a' や 'A' を使ってメッセージに返答するための草稿を用意すると、Emacs が 3 分割されます。上が現在の Summary モード、中が Message モード、下が Draft モードです。

Message モードのテキストを引用するコマンドを以下に示します。

'C-cC-y' Message モードからメッセージの一部をコピーし、引用ラベルと引用記号付でペーストする。

1. おおまかに言えば、Message モードの本文がコピーされる。たとえば、Text/Plain が表示されていると、Message モード全体がコピーされる。Message/Rfc822 が表示されている場合は、ヘッダを除いた本文がコピーされる。
2. 'C-u' と共に呼ばれると、ヘッダがあればヘッダをコピーする。
3. Emacs のマークがあると、そのマークとカーソルの間が対象となる。

'C-cC-t' Message モードからメッセージの一部をコピーし、引用ラベルと引用記号なしでペーストする。

デフォルトの引用ラベルと引用記号は以下のようになります。

```
From: SUMIKAWA Munechika <sumikawa@ebina.hitachi.co.jp>
Subject: Wine
Date: Wed, 23 Jul 1997 11:40:50 +0900

> おはようからおやすみまでニートでおなじみの角川です。
>
> さて、とろけるワイン作戦ですが、定石通り '90 のボルドーの
> カベルネ・ソービニョンを狙いたいと思います。ピノノアール
> がちょっぴりブレンドしてあるといいかも。
```

Draft モードでは Message モード ("*mew message*"バッファ) に表示されているものならなんでも引用できます。つまり、複数のメッセージを簡単に引用できるのです。引用したいメッセージを表示させて、本文を引用する手順を、引用したいメッセージの回数だけ繰り返して下さい。そのための 3 分割です。

Mew は supercite とリンクできますが、supercite を利用しようと思う前に、以下のように設定してみて下さい。

```
(setq mew-cite-prefix-function 'mew-cite-prefix-username)
```

この設定をしておくと、以下のように引用記号にユーザ名が付くようになります。

```
From: SUMIKAWA Munechika <sumikawa@ebina.hitachi.co.jp>
Subject: Wine
Date: Wed, 23 Jul 1997 11:40:50 +0900

sumikawa> おはようからおやすみまでニートでおなじみの角川です。
sumikawa>
sumikawa> さて、とろけるワイン作戦ですが、定石通り '89 のボルドーの
sumikawa> カベルネ・ソービニョンを狙いたいと思います。ピノノアール
sumikawa> がちょっぴりブレンドしてあるといいかも。
```

さらに、以下の設定を加えてみましょう。

```
(setq mew-addrbook-for-cite-label 'nickname)
(setq mew-addrbook-for-cite-prefix 'nickname)
```

最初の設定でラベルの中のアドレスがニックネーム (See Section 4.3 [addrbook], page 15) に変わります。また、次の設定で引用記号のユーザ名の部分がニックネームに置き換わります。

```
From: すみっち
Subject: Wine
Date: Wed, 23 Jul 1997 11:40:50 +0900

すみっち> おはようからおやすみまでニートでおなじみの角川です。
すみっち>
すみっち> さて、とろけるワイン作戦ですが、定石通り '89 のボルドーの
すみっち> カベルネ・ソービニョンを狙いたいと思います。ピノノアール
すみっち> がちょっぴりブレンドしてあるといいかも。
```

もし、引用の様式が上記ではなく以下のようになるなら、'`mail-citation-hook`' が定義されているのかもしれません。

```
In article .....
```

Mew 独自の引用様式を使いたいなら、以下の行を ".emacs" に加えて下さい。

```
(setq mail-citation-hook nil)
```

4.6 マルチパートの作成

さて、ここでマルチパートの作り方を披露しましょう。

たとえば、+draft/1 でメッセージを書いているときに、'`C-cC-a`' と入力すると、草稿の一番下に

```
---------------------------- attachments -----------------------------
      Multipart/Mixed                                          1/
     1  Text/Plain(guess)                                      CoverPage*
     2                                                         .
--------0-1-2-3-4-5-6-7-8-9-----------------------------------------
```

という行が挿入されます。"1/" はマルチパートを構築するための一時的なディレクトリで、実体は "~/Mail/draft/mime/1" です。パート 1 の CoverPage は本文を意味します。ここで Draft モードは次のようになっているでしょう。

```
To: mew-dist
Subject: ここがヘッダ
X-Mailer:Mew version 1.94 on XEmacs 20.4
----
本文だよ。

---------------------------- attachments ----------------------------
      Multipart/Mixed                                          1/
     1  Text/Plain(guess)                                      CoverPage*
     2                                                         .
--------0-1-2-3-4-5-6-7-8-9-----------------------------------------
```

3 つの領域を以下のように呼ぶことにします。

- "—-" より上を 「ヘッダ」
- "—-" から "attachments" までを「本文」
- "attachments" より下を「添付領域」

Draft モードでは、リージョンによってキー割当が違います。

たとえば、'TAB' は以下のようになります。

ヘッダ　さまざまな補完。

本文　TAB の挿入。

添付領域　なにもしない。

'c' だと以下のようになります。

ヘッダ　c を挿入。

本文　c を挿入。

添付領域　ファイルのコピー。

以下、添付領域でのキー割当です。

'C-p'　現在のディレクトリの前のファイルへ移動。

'C-n'　現在のディレクトリの後のファイルへ移動。

'C-f'　1 番目のサブディレクトリに移動。

'C-b'　親ディレクトリに移動。

'c'　ファイルのコピー。"." 上で有効。ネットワーク経由でも可。リモートのファイルをコピーする場合は、"/[user@]hostname:/filepath"の形式でファイルを指定。

'l'　ファイルへシンボッリクリンクを張る。"." 上で有効。添付ファイルを 'f' を使って読み込んで編集する場合は、実体を編集してしまわないように、'l' ではなく 'c' でコピーすべき。

'd'　ファイルとディレクトリの消去。

'm'　サブディレクトリ (つまりマルチパート) の作成。"." 上で有効。

'f'　ファイルをバッファに読み込む。

'F'　新規ファイルをバッファに読み込む。"." 上で有効。

‘y’ Message モードに表示されているメッセージにリンクを張る。"." 上で有効。

‘e’ external-body の入力。"." 上で有効。

‘a’ 音をサンプリングしオーディオファイルとして挿入。"." 上で有効。

‘p’ 入力されたユーザの PGP 公開鍵を取り出す。"." 上で有効。

‘D’ ちょっとした説明 (Content-Description:) の入力。

‘T’ データ型 (Content-Type:) の変更。

‘C’ Text/* 型のデータの charset を指定する。

‘P’ 受信側でこのパートを保存する際のファイル名 (Content-Disposition:) の変更。ファイル名の入力の際に、単に ‘RET’を押すと値が消える。そして、送信側のファイル名が ‘*’ と共に表示される。

添付領域では、ファイルのサフィックスによってデータを取り扱います。現在サポートしているサフィックスは以下の通りです。

```
.txt        Text/Plain
.html       Text/Html
.rfc822     Message/Rfc822
[0-9]+      Message/Rfc822
.ext        Message/External-body
.ps         Application/PostScript
.tar        Application/Octet-stream ;; dummy
.gif        Image/Gif
.jpg        Image/Jpeg
.jpeg       Image/Jpeg
.png        Image/Png
.xwd        Image/X-xwd
.xbm        Image/X-xbm
.bmp        Image/X-bmp
.au         Audio/Basic
.mpg        Video/Mpeg
.mpeg       Video/Mpeg
.pgp        Application/Octet-Stream
.pka        Application/Pgp-keys
.*          Text/Plain
```

‘c’ でファイルをコピーすると、たとえば次のようになります。コピーするときのファイル名は、適切なデータ型を推測できるようサフィックスに気を付ければなんでもよいです。

```
---------------------------- attachments ----------------------------
      Multipart/Mixed                                          1/
     1  Text/Plain(guess)                                      CoverPage*
B    2  Image/Gif                    MagicPoint のロゴ         mgp.gif
Q    3  Application/Postscript       資料                      ohp.ps
     4                                                         .
--------0-1-2-3-4-5-6-7-8-9-----------------------------------------
```

各行は、

– マーク (Content-Transfer-Encoding:)

- パート番号
- データの型 (Content-Type:)
- 説明 (Content-Description:)
- ファイル名 (Content-Disposition:)

から構成されています。

マーク (Content-Transfer-Encoding:) を変更する方法は、See Section 4.11 [mark-b-comp], page 27 を参照して下さい。データの型 (Content-Type:) は 'T' によって変えられます。説明 (Content-Description:) は 'D' で入力できます。この説明のカラムは、See Section 4.11 [mark-b-comp], page 27 で説明する暗号化の際に上書きされます。

第 5 カラムに表示されるのは、実際にはコピーしたファイル名か Content-Disposition:、つまり、受信者がそのパートを保存する際のファイル名です。Content-Disposition: の値があれば、それが表示されます。なければ、コピーしたファイル名に '*' を付加して表示します。ファイルをコピーした際の Content-Disposition: の値は、コピーしたファイル名が指定されています。ただし、Message/* と Multipart/* には Content-Disposition: は設定されません。Content-Disposition: を指定するには、'P' を利用して下さい。

ファイルはシングルパートに、ディレクトリはマルチパートに対応します。ですから、ファイル構造を作っていく感覚で複雑なマルチパートを作成できます。簡単でしょ？

ディレクトリのデフォルトの Content-Type: は Multipart/Mixed です。これも 'T' によって変更できます。

さて、お好みのマルチパートが作成できたら、'C-cC-m' とタイプしましょう。あーーーら不思議。ファイル構造がマルチパートへ変換されるではありませんか。むろん、多段のマルチパートもサポートしています。あとは、'C-cC-c' で送るだけです。

MIME の文法が分かっていない人は、'C-cC-m' の後に草稿を変更しないようにしましょう。もし、どうしても草稿を修正するときは、「最初の境界の前と最後の境界の後は無視される」ことに注意して下さい。

マルチパートの変換後、やっぱり元に戻したいと思ったら、'C-cC-u' を使って下さい。('C-xu' や 'C-_' ではないことに注意。)

パートの実体が外部にある external-body を作成するコマンド 'e' について説明しましょう。access-type に ftp か anon-ftp を入力するときは、ange-ftp のおかげでリモートのファイル名が補完できます。access-type が local-file の場合は、もちろんファイル名を補完できます。

もし、マルチパートの作成途中でやっぱりシングルパートに戻したくなったら、一番上のマルチパート部分で 'd' を押して下さい。

4.7 文字コードの推測

Mew はシングルパートとマルチパートの両方に対し、charset を推測する機能を持っています。

<シングルパート>

Draft モードで 'C-cC-m' と入力すると、Mew は本文の charset を推測します。Bilingual Emacs では、7 ビットの文字コードに対し US-ASCII を選び、8 ビットの文字コードに対し ISO-8859-1 を選択します。Mule では、Mule で定められた文字コードの内部表現から charset を推測します。

<マルチパート>

テキストファイルを添付領域に添付した際には、以下のように "(guess)" と表示されます。

```
----------------------------- attachments -----------------------------
      Multipart/Mixed                                             1/
     1  Text/Plain(guess)                                          CoverPage*
     2  Text/Plain(guess)                                          textfile
     3                                                             .
--------0-1-2-3-4-5-6-7-8-9-----------------------------------------------
```

シングルパートとマルチパートで charset を推測することの違いは、データがどこに存在するかです。シングルパートはバッファに格納されていますが、マルチパートのそれぞれのファイルはディスク上に存在します。charset を推測するためには、Mew はこれらのファイルをバッファに読み込み、そしてシングルパートと同じ要領で文字コードを推測します。

Bilingual Emacs では、Mew はファイルをそのままの形式で読み込みます。ですから、7 ビットのファイルには US-ASCII が、8 ビットのファイルには ISO-8859-1 が選ばれます。

Mule では、Mew はファイルをそのサイトの環境 (つまり、auto conversion) に従ってファイルを読み込みます。この環境はサイトごとによって異なります。日本語の環境では、Mule は 2022-JP、EUC-Japan、そして、Shift_JIS を見事に推測し、日本語用の内部表記に変換してバッファに格納します。Mew は、この内部表記から charset を推測します。よって、ISO-2022-JP、EUC-Japan、そして、Shift_JIS のファイルを安全に添付できます。Mule でこの環境を決定する関数は、set[up]-<language>-environment という名前ですから、詳しいことが知りたいならこれらの関数の説明を読んで下さい。

もし、ファイルの charset を明示的に指定したいなら、'C' を使って下さい。典型的な使用例は、日本で ISO-8859-1 のファイルを添付することです。この例の場合、添付領域は以下のようになります。

```
----------------------------- attachments -----------------------------
      Multipart/Mixed                                             1/
     1  Text/Plain(guess)                                          CoverPage*
     2  Text/Plain(iso-8859-1)                                     textfile
     3                                                             .
--------0-1-2-3-4-5-6-7-8-9-----------------------------------------------
```

Bilingual Emacs では 'C' は利用できません。

4.8 メッセージへの返答と宛先の決定

新規にメッセージを書く場合は、To: や Cc: を自分で書くことになります。一方 Summary モードで 'a' や 'A' を使って、あるメッセージに返答しようとすると、To: や Cc: は自動的に用意されます。

返答の際、Mew は以下のような手順に従って To: や Cc: を用意します。

返答するメッセージの From: が自分以外の場合:

返答するメッセージに Reply-To: がない場合:

返答するメッセージの From: を To: へ (1)

返答するメッセージの To: と Cc: を Cc: へ (2)

返答するメッセージに Reply-To: がある場合:

返答するメッセージの From: と Reply-To: を To: へ (3)

返答するメッセージの To: と Cc: を Cc: へ (4)

返答するメッセージの From: が自分である場合:

返答するメッセージの To: を To: へ (5)

返答するメッセージの Cc: を Cc: へ (6)

ただし、あるアドレスが複数ある場合は、自動的に 1 つになります。また、匿名の宛先を表す ":;" で終るアドレスも、自動的に消去されます。

自分のアドレスは自動的に消去されます。自分のアドレスを定義するには 'mew-mail-address-list' を使います。以下に例を示します。

```
(setq mew-mail-address-list
      '("pooh@[a-z]*.aist-nara.ac.jp"
        "pooh@mew.org"
        "winnie@iijlab.net"))
```

(1) 〜 (6) でどのフィールドをコピーするかは、以下の変数で指定できます。

(1) 'mew-noreplyto-to-list'

(2) 'mew-noreplyto-cc-list'

(3) 'mew-replyto-to-list'

(4) 'mew-replyto-cc-list'

(5) 'mew-fromme-to-list'

(6) 'mew-fromme-cc-list'

Reply-To: がある場合に、Reply-To: だけに返答したいと思うなら、以下のように設定すればよいでしょう。

```
(setq mew-replyto-to-list '("Reply-To:"))
(setq mew-replyto-cc-list nil)
```

'a' や 'A' を 'C-u' 付で呼び出すと、返答するメッセージの From: が To: に入り、Cc: は空になります。送信者のみに返答する場合に利用します。

4.9 メッセージの転送

メッセージを転送するには、Summary モードで 'f' や 'F' を利用します。すると、Draft モードに移行し、あらかじめメッセージが添付領域に添付された草稿が準備されます。

また Draft モードで添付領域を用意し、メッセージをコピー ('c') したりメッセージにリンク ('l') を張ったりしても、メッセージを転送できます。ファイル名が数字 ([0-9]+) の場合は、自動的にメッセージだと判断されます。また、添付領域で 'y' を使うと、Message モードに表示しているメッセージにリンクを張るので便利です。

通常は添付したメッセージの全体が転送されます。もし、ヘッダの一部を削りたい場合は、'mew-field-delete-for-forwarding' を定義して下さい。以下に "Received:" と "Return-Path:" を転送時に削るための設定例を示します。

```
(setq mew-field-delete-for-forwarding '("Received:" "Return-Path:"))
```

4.10 PGP を利用する

ここでは、テキストである本文を PGP で署名したり暗号化したりする方法について説明します。出てくるコマンドは以下の通りです。

'C-cC-s' 草稿全体を PGP で署名する。パスフレーズを入力すること。

'C-cC-e' 草稿全体を PGP で暗号化する。

'C-cC-b' 草稿全体を PGP で署名後暗号化する。パスフレーズを入力すること。

'C-cC-r' 草稿全体を PGP で暗号化後署名する。パスフレーズを入力すること。

メッセージを暗号化するには受信者の公開鍵を使用します。逆に署名するには自分の秘密鍵を使います。よって、署名するためにはパスフレーズを入力する必要があります。ただし、パスフレーズの保存機能を使っており、パスフレーズが保存されている場合は、パスフレーズを入力する必要はありません (See Section 3.3 [pgp-viewing], page 6)。

これらは、次節で説明するマークを使った PGP/MIME の作成方法の省略方法に当たります。

Mew で PGP を使うためには、PGP の userid としてアドレスを選ぶ必要があります (例 "Kazuhiko Yamamoto <kazu@mew.org>")。

以降の説明では、次の例を取り上げます。

```
To: pooh
Subject: PGP/MIME を使おうよ
X-Mailer:Mew version 1.94 on XEmacs 20.4
----
Mew がセキュリティ・マルチパートをサポートしました。

--かず
```

署名するには、'C-cC-s' と入力します。すると、次のメッセージが得られます。

```
To: winnie-the-pooh@100acre.woodwest.uk
Subject: PGP/MIME =?iso-2022-jp?B?GyRCJHI7SCQqJCYkaBsoQg==?=
X-Mailer:Mew version 1.94 on XEmacs 20.4
Mime-Version: 1.0
Content-Type: Multipart/Signed;
        protocol="application/pgp-signature";
        micalg="pgp-md5";
        boundary="--Security_Multipart(Sat_Nov_16_03:55:00_1996)--"
Content-Transfer-Encoding: 7bit

----Security_Multipart(Sat_Nov_16_03:55:00_1996)--
Content-Type: Text/Plain; charset=iso-2022-jp
Content-Transfer-Encoding: 7bit

Mew がセキュリティ・マルチパートをサポートしました。

--かず

----Security_Multipart(Sat_Nov_16_03:55:00_1996)--
Content-Type: Application/Pgp-Signature
```

```
Content-Transfer-Encoding: 7bit

-----BEGIN PGP MESSAGE-----
Version: 2.6.3i

iQCVAwUAMoy8ig9kihyeT3RNAQHt7AQAySDg4n8pOp/YuLaAp68Un/YDtWS0FnOC
7EqHJd6fyViPBnZq8d+uGikA7kOBTz+8Kcv+hN6I7BrQVJGEzd0Y9yHHhXvZj++1
0DO9vgWL5G/Zfk/JMnLBt/BZ1ppOhJPT/L5qi2abk+mBVMKxQe071lfFEfvjF1C2
8trTXm/bBz4=
=TvAG
-----END PGP MESSAGE-----

----Security_Multipart(Sat_Nov_16_03:55:00_1996)----
```

'C-cC-c' で送信して下さい。

暗号化するには、'C-cC-e' と入力して下さい。次のようになります。

```
To: winnie-the-pooh@100acre.woodwest.uk
Subject: PGP/MIME =?iso-2022-jp?B?GyRCJHI7SCQqJCYkaBsoQg==?=
X-Mailer:Mew version 1.94 on XEmacs 20.4
Mime-Version: 1.0
Content-Type: Multipart/Encrypted;
        protocol="application/pgp-encrypted";
        boundary="--Security_Multipart(Sat_Nov_16_03:57:47_1996)--"
Content-Transfer-Encoding: 7bit

----Security_Multipart(Sat_Nov_16_03:57:47_1996)--
Content-Type: Application/Pgp-Encrypted
Content-Transfer-Encoding: 7bit

Version: 1

----Security_Multipart(Sat_Nov_16_03:57:47_1996)--
Content-Type: Application/Octet-Stream
Content-Transfer-Encoding: 7bit

-----BEGIN PGP MESSAGE-----
Version: 2.6.3i

hIwDD2SKHJ5PdE0BA/9gUkcQYVfT+3LrUmcgLkNepuOnDfjADHrWiNo10t4ijyf8
ODBpUBXoBdTg08eNLAWmRFhiJPmI+mxpF6cYFZXhr7gVpa0Qzp3Gr9nYvngRPKNK
qUiQjA/ORR3c1TBawufB19jJ9RdU2f0BidhzOSbzsJh1LTgUZu/7Qyd02LxyEqYA
AACbrV867PeoFyFc9MVfqTUR6Zw6kGBAlnVYjqQgBhuuyG79vbAbDJMhFiRpoRPf
0MqEewxRonwKOik/PoKnLrwFg77Cb5pxRqMiWPyECJnqtX7r7Wg1c8kqPD0VRjI9
GhHPiG/RmNbpbj/5g6zZri1YBCe8qxIS0QKa3YO7HRDcdBFARr22RaFGFtgdBQ6X
cZB+qNeEaKXt3AneTWc=
=djCr
-----END PGP MESSAGE-----

----Security_Multipart(Sat_Nov_16_03:57:47_1996)----
```

‘C-cC-c’ で送信して下さい。このメッセージは、受信者の公開鍵に加えて、自分の公開鍵でも暗号化されています。ですから、保存したメッセージを復号化できます (例 ‘g’ で +Backup に移動したとき)。

署名後暗号化するには、‘C-cC-b’ と入力します。暗号化後署名するには、‘C-cC-r’ とタイプします。いずれの場合も、‘C-cC-c’ で送信して下さい。

PGP で署名を施したり、暗号化したりしてメッセージを送ろうと思っていても、うっかり忘れることがあります。そのため、メッセージを作成するコマンド ‘C-cC-m’ に対し、必要に応じて PGP を起動させる機能があります。

作成するすべてのメッセージのプライバシを保護したいなら、‘mew-protect-privacy-always’ を ‘t’ にして、‘mew-protect-privacy-always-type’ に利用したいサービスを設定します。

暗号化されたメッセージに対する返答メッセージのプライバシを保護したいなら、‘mew-protect-privacy-encrypted’ を ‘t’ にして、‘mew-protect-privacy-encrypted-type’ に利用したいサービスを設定します。この設定は、暗号化されたメッセージへの返答する場合、上記のすべてのメールに対する設定よりも優先されます。

以下に利用できるサービスを示します。かっこ内はそれぞれのサービスを表すシンボルです。‘C-cC-m’ した際に利用されるサービスは、モードラインに表示されます。

pgp-signature (PS)
: 署名

pgp-encryption (PE)
: 暗号化

pgp-signature-encryption (PSPE)
: 署名後暗号化

pgp-encryption-signature (PEPS)
: 暗号化後署名

以下の例は、すべてのメールに対し ‘C-cC-m’ で署名する設定です。

```
(setq mew-protect-privacy-always t)
(setq mew-protect-privacy-always-type 'pgp-signature)
```

以下の例は、暗号化されたメッセージへの返答メッセージに対し、‘C-cC-m’ で暗号化する設定です。

```
(setq mew-protect-privacy-encrypted t)
(setq mew-protect-privacy-encrypted-type 'pgp-encryption)
```

Draft モードでは、‘C-cC-pC-a’ で ‘mew-protect-privacy-always’、‘C-cC-pC-e’ で ‘mew-protect-privacy-encrypted’ の値を反転できます。

Draft モードにおいて現在書いている草稿対してのみあらかじめサービスを指定しておき、送信時にサービスを施すことを忘れないようにできます。現在の草稿に対し ‘C-cC-m’ で施されるサービスを指定するには、‘C-cC-pC-d’ に続いて上記のサービスの 1 つを入力して下さい。

4.11 マークを使った PGP/MIME の作成

PGP/MIME をサポートするために、マークを使った作成方法が提供されています。以前の例を思い出してみましょう。

```
---------------------------- attachments ----------------------------
      Multipart/Mixed                                          1/
     1  Text/Plain(guess)                                      CoverPage*
B    2  Image/Gif                    MagicPoint のロゴ         mgp.gif
Q    3  Application/Postscript       資料                      ohp.ps
     4                                                         .
--------0-1-2-3-4-5-6-7-8-9-------------------------------------------
```

行頭に ‘B’ や ‘Q’ といったマークがあります。このマークは符号化を意味しています。Mew では、新しい概念「符号化」を導入しています。符号化には、Base64, Quoted-Printable, Gzip64(Gzip + Base64), PGP で署名, PGP で暗号化などがあります。

現在次の 6 つのマークがサポートされています。

‘" "’ 符号化しない。ただし、8 ビットのテキストは符号化されるかもしれない。

‘B’ Base64

‘Q’ Quoted-Printable

‘G’ Gzip64(gzip 圧縮し Base64 で符号化する。Mew が実験的に採用している。相手が Mew を使っていない場合は、使用すべきではない。)

‘PS’ PGP で電子署名。

‘PE’ PGP で暗号化。

添付領域でのマークに関係する新しいキー割当は以下の通りです。

‘B’ Base64 で符号化するため ‘B’ マークを付ける。

‘Q’ Quoted-Printable で符号化するため ‘Q’ マークを付ける。

‘G’ Gzip64 で符号化するため ‘G’ マークを付ける。ただし、Text/Plain と Application/Postscript でしか実行できない。これ以外の型には圧縮は無意味である。なぜなら、jpeg などはあらかじめ圧縮されているから。

‘S’ PGP で署名するため ‘PS’ マークを付ける。

‘E’ PGP で暗号化するため ‘PE’ マークを付ける。受信者のアドレスを入力する。

‘U’ 符号化を元に戻す。元々のマークに戻る。

次の例を考えてみましょう。パート 2 は PGP で署名され、"kazu" 用に PGP で暗号化されます。安心して下さい。説明の部分は上書きされていますが、保存されています。パート 3 は Gzip64 で符号化されます。

```
---------------------------- attachments ----------------------------
      Multipart/Mixed                                          1/
     1  Text/Plain(guess)                                      CoverPage*
PSPE 2  Image/Gif                    kazu@mew.org              mgp.gif
G    3  Application/Postscript       資料                      ohp.ps
     4                                                         .
--------0-1-2-3-4-5-6-7-8-9-------------------------------------------
```

適宜マークを付けた後は、‘C-cC-m’ で MIME(PGP/MIME など) を作成し、‘C-cC-c’ で送信して下さい。

4.12 PGP の鍵の配布

PGP の公開鍵を配布するには、Draft モードの添付領域で 'p' を押して下さい。だれの公開鍵を配布するか尋ねられます。自分の公開鍵であれば、単に 'RET' と入力して下さい。他人の公開鍵であれば、補完を利用しながらその人のアドレスを入力して下さい。PGP の公開鍵は、Application/Pgp-keys というデータ型で配送されます。

Summary モード、あるいは、Virtual モードでメッセージを読んでいる際に、あるパートのデータ型が Application/Pgp-keys であれば、Mew は PGP の公開鍵リングにそれを登録しようとします。Mew は、「信用度」と「有効性」を全く考慮しないことに注意して下さい。これらの値を設定するのは、あなた自身です。設定には、"pgp -ke" と "pgp -ks" を使って下さい。もし、「信用度」と「有効性」の意味が分からなければ、PGP を使って自分のプライバシを保護しようとする前に、PGP が提供する「信用の輪」とは何かを学ぶべきです。

5 愉快なマークたち

ここでは、Summary モードで利用できるマークについて説明します。マークは以下のように、数字の右に付きます。

```
1D 07/17 いとぢゅん     v6: items to be no in6_pcbnotify() がなにも
2o 07/18 歌代先生       Re: behavior after これ、mark-ring がどんど
3* 07/19 のむさん       refile info.       乃村です。遅くなりました。
```

現在利用できるマークは以下の 4 つです。

‘D’ 消去のマーク。

‘o’ 整頓、つまり、フォルダを移動させるマーク。

‘@’ 複数のメッセージを一度に扱うためのマーク。

‘*’ 後から読み返すためのマーク。

以下それぞれについて説明します。

5.1 消去 ‘D’

メッセージを消去するには、まず Summary モードで ‘d’ を押して、‘D’ マークを付けます。マークを付けただけでは何も起こらないので、間違って ‘d’ を押しても大丈夫です。デフォルトでは、‘x’ を押すと ‘D’ マークの付いたメッセージが +trash フォルダに移動します。

+trash フォルダ内のメッセージを実際に消去するには、デフォルトでは以下の 2 つの方法があります。

1. Summary モードで ‘D’ を実行する。
2. +trash フォルダ内で ‘D’ マークを付けて、‘x’ を押す。

これまでしつこいように「デフォルトでは」と念を押してきました。ここでいうデフォルトとは、‘mew-msg-rm-policy’ が ‘’trashonly’ であるということです。‘mew-msg-rm-policy’ は以下のような値をとれます。それぞれの説明は ‘x’ を押した際の動作についてです。

‘’totrash’
: +trash フォルダ以外のフォルダでは、‘D’マークの付いたメッセージが+trash フォルダに移動します。+trash フォルダでは単にマークが消えます。

‘’always’ ‘D’ マークの付いたメッセージは実際に消去されます。

‘’trashonly’
: +trash フォルダの ‘D’ マークの付いたメッセージは、実際に消去されます。それ以外のフォルダにある ‘D’ マークの付いたメッセージは、+trash フォルダに移動します。

‘’uselist’
: ‘mew-msg-rm-folder-list’ で指定されたフォルダの ‘D’ マークの付いたメッセージは、実際に消去されます。それ以外のフォルダにある ‘D’ マークの付いたメッセージは、+trash フォルダに移動します。

‘それ以外’ ‘’totrash’ と同様に扱われます。

自分の好きなように 'x' の動作をカスタマイズして下さい。

すべての '*' マークを 'D' に変換できれば、一度にたくさんの 'D' マークを付けれて便利です。これには、'md' を利用して下さい。

以下に、'D' マークに関するコマンドをまとめます。

'd' 'D' マークを付ける。

'x' 'D' マークの付いたメッセージを 'mew-msg-rm-policy' に従って処理する。

'md' '*' マークすべてを 'D'マークに変換する。

5.2 整頓 'o'

メッセージを整頓するには 'o' を押して、整頓先のフォルダを入力し、'o' マークを付けます。整頓先のフォルダは、賢く推測してくれるので、ほどんどの場合はフォルダ名を入力する代わりに 'RET' を押すだけです。"," で区切って複数のフォルダを入力することもできます。もちろん、'TAB' で補完できます。詳しくは See Chapter 6 [Refile], page 34 を参照して下さい。

'o' マークの付いたメッセージの上で 'o' を押すと、整頓先を追加したり変更したりできます。また、実際の整頓は 'x' と入力されたときに実行されます。

以下に、'o' マークに関するコマンドをまとめます。

'o' 'o' マークを付ける。

'x' 'o' マークの付いたメッセージを整頓する。

'mo' '*' マークの付いたメッセージに対し入力されたフォルダへ整頓するための 'o' マークを付ける。

5.3 複数 '@'

複数のメッセージを一度に取り扱うためには、'@' マークを付けます。'@' マークが付いた単数／複数のメッセージを取り扱うコマンドは以下の通りです。

'@' '@' マークを付ける。

'F' '@' マークの付いたメッセージを MIME 形式で転送するための草稿を準備。

'M-s' '@' マークの付いたメッセージを "unshar" の入力として渡す。つまり、shar (の後に split) されたデータを取り出す。

'M-t' '@' マークの付いたメッセージを "uumerge" の入力として渡す。つまり、uuencode(の後に split) されたデータを取り出す。

'M-b' '@' マークの付いたメッセージに格納されているメッセージを取り出す。

'J' 大きなメッセージは Message/Partial として複数に分割されている場合がある。このコマンドは、'@' マークの付いた Message/Partial のメッセージから元のメッセージを生成する。

'M-s' や 'M-t' では、'@' マークの付いたメッセージの番号がきちんと分割順になっている必要があります (番号はとびとびでも構いません)。ちゃんと順番になってないなら、'S' でソートするとよいかもしれません。

5.4 復習 '*'

後から読み返したいメッセージには、'*' で '*' マークを付けて下さい。また、選択コマンド '?' を使うと、入力した条件にマッチするメッセージに '*' マークが付きます (詳しくは、See Chapter 7 [Pick], page 40 を参照して下さい)。'N' や 'P' で '*' マークの付いたメッセージに移動し表示できます。

以下に、'*' に関連するコマンドを示します。

'*' '*' マークを付ける。

'N' 下方向の '*' マークの付いたメッセージへ移動し表示。

'P' 上方向の '*' マークの付いたメッセージへ移動し表示。

'ma' マークの付いていないメッセージすべてに '*' マークを付ける。

'mr' 入力した正規表現にマッチしたメッセージに '*' マークを付ける。

'md' '*' マークを 'D' マークに変換。選択コマンド '?' で選んだメッセージを消去するときに便利。

'mo' '*' マークを 'o' マークに変換。選択コマンド '?' で選んだメッセージをあるフォルダに整頓する場合に便利。

5.5 マークの消去

'o' マークや 'D' マークが付いたメッセージは、マーク実行コマンド 'x' を押さない限り処理されません。よって、'x' を押す前に、マークを取り止めるコマンド 'u' でマークを消せば、誤ってメッセージを消すことはありません。

以下にマークを消去するコマンドをまとめます。

'u' 現在のメッセージのマークを消す。

'U' 入力したマークが付いているすべてのメッセージのマークを消す。

5.6 マークの強さ

マークには「強いマーク」と「弱いマーク」があります。同じレベルのマークは上書きできます。強いマークは弱いマークを上書きできます。

マークを付けた際の動作は、以下の通りです。

強いマーク :: 'o' と 'D'
: 新たにマークしたときは、次のメッセージを表示。上書きした場合は、その行に留まる。

弱いマーク :: '*' と '@'
: 常にその行に留まる。

強いマークを付けた後にカーソルが動く方向については、See Section 9.1 [level-one], page 43 を参照して下さい。

マークは以下のように交換できます。

'm@' '*' -> '@' :: 選択コマンド '?' で選び、"uumerge" を起動するコマンド 'M-t' を利用する場合に便利。

‘m*’ ‘@’ -> ‘*’

‘ms’ ‘@’ <-> ‘*’

‘md’ ‘*’ -> ‘D’ :: 選択コマンド ‘?’ で選んだメッセージを消去するときに便利。

‘mo’ ‘*’ -> ‘o’ :: 選択コマンド ‘?’ で選んだメッセージをあるフォルダに整頓する場合に便利。

6 楽々整理整頓

1 日に数百のメッセージを受け取るようになると整理整頓が大変になります (え、そんなに受け取らないですって？ 幸せですね :) 。Mew では、‘o’ でメッセージを整頓する際に、整頓先を推測しデフォルト値として表示してくれます。たとえば、次のようになります。

```
Folder name (+work/mew-dist): +
```

もし、() の中のデフォルト値が自分の希望通りであれば、‘RET’ を押すだけでよいのです。整頓先が決定しているメッセージには、‘o’ マークが付きます。

この整頓先の推測が賢ければ賢い程ユーザは楽になります。Mew では以下のようなルールが用意されています。

6.1 メーリングリスト用のフォルダから推測

あるメーリングリスト宛のメッセージを、そのメーリングリスト名のフォルダに整理することは多いと思います。Mew ではメーリングリスト宛に届いたメッセージに対して、それ用のフォルダを推測する機能があります。

たとえば、+misc/pooh-lovers というフォルダがあったとしましょう。次のようなメッセージは、このフォルダに整頓すればよい可能性が高いといえます。

```
To: pooh-lovers@mew.org
```

このように、To: や Cc: のアドレスが、フォルダ名の一番右側にマッチするものがないか探すわけです。フォルダを階層化していない人が多いようですが、Mew を使う限り、階層化しない手はありません。

さて、鋭い人は次のように個人のアドレスが To: や Cc: にある場合、困るのではないかと思うでしょう。

```
To: piglet@mew.org
Cc: pooh-lovers@mew.org
```

たとえば、pooh は pooh-lovers の一員ですから、このメッセージが届きます。しかし、piglet と仲がいいので、+from/piglet にマッチしてしまいます。

そこで、Mew では無視するフォルダを設定できるようになっています。デフォルトでは、+from 以下を無視します。ですから、個人からのメッセージは +from 以下にに収めて下さい

候補が決定できたら

```
Folder name (+misc/pooh-lovers): +
```

と訊いてきます。あっていれば ‘RET’ を、違っていればお望みのフォルダを入力して下さい。

‘o’ で新しいフォルダを指定すると、そのフォルダが自動的に作成され、次からは推測用の候補にも加わります。便利でしょ？

この機能を提供する関数は ‘mew-refile-guess-by-folder’ です。

6.2 指定したルールから推測

フォルダ名から推測する機能だけでは、思うようなフォルダを推測してくれない場合があります。たとえば、To: が staff@mew.org であるメッセージと To: が staff@iijlab.net であるメッセージに対し、フォルダ名からの推測では同じフォルダ (たとえば、"+net/staff") が選ばれてしまいます。そこで、Mew では、変数 ‘mew-refile-guess-alist’ に明示的にルールを設定できます。

1 つ例を挙げてみましょう。

```
(setq  mew-refile-guess-alist
  '(("To:"
      ("staff@mew.org" . "+net/mew/staff")
      ("staff@iijlab.net" . "+net/iijlab/staff")
     )))
```

これは、メッセージヘッダ中の To: の横の文字列に staff@mew.org があれば+net/mew/staff へ、staff@iijlab.net があれば +net/iijlab/staff へ整頓するという意味です。

ルールは、以下のように書きます。

```
rule ::= '((<key> <alist>) (<key> <alist>) (<key> <alist>) ...)
```

全体は (<key> <alist>) のリストです。<key> はフィールド名を書きます。<alist> は以下のようになります。

```
<alist> ::= (<value> . <folder>|<rule>) (<value> . <folder>|<rule>) ...
```

<value> は <key> で示したフィールドにくる値です。<folder> は <key> にマッチした際にどのフォルダに整頓するかを意味しています。<value> と <folder> を '.' で区切るのを忘れないで下さい。<folder> の代りに <rule> を再帰的に記述することもできます。

特殊な <key> として 'nil' と 't' があります。'nil' は、何も推測できなかった場合に返す <value> を指定するために用います。't' は、推測した値に加えて返す <value> を指示するために使います。

正規表現を知っている人は、以下のような複雑なルールを設定できます。

```
(setq mew-refile-guess-alist
  '(("Newsgroups:"
    ("^nifty\\.\\([^ ]+\\)" . "+Nifty/\\1")
    (".*"          . "+rec/news"))
   ("To:"
    ("\\(inet\\|wide\\)@wnoc-fuk" . "+wide/\\1-wnoc-fuk"))
   ("From:"
    ("uucp@"       . "+adm/uucp")
    ("ftpsync@"    . "+adm/ftpsync"))
   (nil . "+unknown")))
```

この機能を提供する関数は 'mew-refile-guess-by-alist' です。

6.3 対話関係から推測

Mew には、整頓しようとしているメッセージの親のメッセージが以前整頓されたフォルダを選択してくれる機能があります。

たとえば、pooh、piglet、roo との間で、蜂蜜を取りに行こうという話題が盛り上がったとしましょう。pooh は、+project/honey というフォルダを作って、最初のメッセージをそこに 整頓したとしましょう。以降、3 人の間のメッセージがきちんとした返答であるかぎり、+project/honey を推測してくれます。

あるメッセージをどこに保存したかという情報は、"~/Mail/.mew-refile-msgid-alist" に保存されています。この情報を過去何通のメッセージに関して保存するかは、'mew-lisp-max-length' で決定します。デフォルトは 1000 通です。2000 通にしたい場合は "~/.emacs" 中で以下のように設定して下さい。

```
(setq mew-lisp-max-length 2000)
```

この機能を提供する関数は 'mew-refile-guess-by-thread' です。

6.4 個人用のフォルダから推測

See Section 6.1 [by-folder], page 34 で説明したメーリングリスト用のフォルダを推測するに加えて、個人用のフォルダを推測する機能があります。個人用のフォルダは +from 以下にありますから、+from 以下のフォルダを選択する機能だともいえます。以下の例を考えてみましょう。

```
To: pooh@mew.org
From: piglet@mew.org
```

piglet から pooh にメッセージが来ました。pooh がこの機能を使うと、From: を手がかりに +from/piglet が選択されます。(+from 以下は階層化されていても構いません。また、フォルダ名はユーザ名だけではなくアドレス全体でも OK です。)

この機能を提供する関数は、‘mew-refile-guess-by-from-folder’ といいます。

次に、pooh が piglet に返答した場合を考えましょう。pooh は自分自身に Cc: していたので、自分にメッセージが戻ってきました。

```
To: piglet@mew.org
Cc: pooh@mew.org
From: pooh@mew.org
```

pooh の立場になって考えてみて下さい。このメッセージを +from/pooh に整頓するか、あるいは +from/piglet に整頓するのかは、好みが分かれるところでしょう。そこで、どちらを選択するのかカスタマイズできるようになっています。

‘mew-refile-guess-from-me-is-special’ が ‘t’ なら、‘mew-refile-guess-by-from-folder’ は、From: が自分のアドレスの場合に、To: と Cc: にあるアドレスをもとに、+from 以下のフォルダを選択します。

6.5 From: から推測

From: に同一のアドレスを持つメッセージが、かつてどこに整頓されたかによって推測する機能があります。

たとえば、piglet は piglet@beech.tree.uk と p-p-p@mew.org の 2 つのアドレスを持っているとしましょう。どちらのアドレスからメッセージが届いても、pooh はそれらを +from/piglet に整頓したいと思っています。もちろん、以下のように明示的にルールを書けば実現できます。

```
(setq  mew-refile-guess-alist
  '(("From:"
      ("piglet@beech.tree.uk" . "+from/piglet")
      ("p-p-p@mew.org"        . "+from/piglet"))))
```

しかし、いちいちルールを書くのは面倒です。そこで、まず From: が piglet@beech.tree.uk であるメッセージを +from/piglet に整頓します。これでフォルダ +from/piglet が作成されます。次に、From: が p-p-p@mew.org であるメールを +from/piglet に整頓したとします。ここで Mew は、p-p-p@mew.org が +from/piglet に整頓されたことを学習します。以後 From: が p-p-p@mew.org であるメッセージを整頓しようとすると、+from/piglet を選択するようになります。

その他、機械からくるメッセージは、いつも +adm/misc に入れることにしたい場合なども、明示的なルールを書かずに済ませられます。

From: とフォルダの情報は、"~/Mail/.mew-refile-from-alist" に保存されています。この情報を過去何通のメッセージに関して保存するかは、See Section 6.3 [by-thread], page 35 と同様に ‘mew-lisp-max-length’ で決定します。

この機能を提供する関数は ‘mew-refile-guess-by-from’ です。

‘mew-refile-guess-from-me-is-special’ が ‘t’ の場合、‘mew-refile-guess-by-from’ は ‘mew-refile-guess-by-from-folder’(See Section 6.4 [by-from-folder], page 36) と同様な動きをします。

6.6 Newsgroups: から推測

ネットニュースをメッセージで受けて Mew で読んでいる人のために、Newsgroups: からフォルダを推測する機能を用意しました。将来 Mew がネットニュースをサポートした場合にも有効でしょう。関数名は ‘mew-refile-guess-by-newsgroups’ です。

6.7 デフォルトの規則

デフォルトの規則は、From: からアドレス名を切り出して、‘+from/user@domain’ を選ぶようになっています。ただし、‘mew-refile-guess-strip-domainpart’ が ‘t’ ならユーザ名を切り出すので、‘+from/user’ が選択されます。

関数名は、‘mew-refile-guess-by-default’ です。

6.8 ルールの制御

Mew では、フォルダ推測のルールを 2 つの変数、‘mew-refile-guess-control’ と ‘mew-refile-ctrl-multi’ で制御します。‘mew-refile-guess-control’ は、呼び出す関数を順に定義します。候補を複数にしたい場合は ‘mew-refile-ctrl-multi’ を ‘t’ に、単数にしたい場合は ‘nil’ に設定します。

標準では、‘mew-refile-guess-control’ は以下のように宣言されています (宣言なので ‘defvar’ が使われています)。

```
(defvar mew-refile-guess-control
  '(mew-refile-guess-by-alist
    mew-refile-ctrl-throw
    mew-refile-guess-by-newsgroups
    mew-refile-guess-by-folder
    mew-refile-ctrl-throw
    mew-refile-ctrl-auto-boundary
    mew-refile-guess-by-thread
    mew-refile-ctrl-throw
    mew-refile-guess-by-from-folder
    mew-refile-ctrl-throw
    mew-refile-guess-by-from
    mew-refile-ctrl-throw
    mew-refile-guess-by-default))
```

Mew は ‘mew-refile-guess-control’ に並べられた関数を順番にすべて実行します。各々の関数が複数の候補を推測することがあります。

‘mew-refile-guess-control’ の動作例として以下を考えてみましょう。

‘mew-refile-guess-by-alist’
が +aaa, +bbb を推測。

‘mew-refile-guess-by-folder’
が +ccc, +ddd を推測。

‘mew-refile-guess-by-default’
が +eee を推測。

+aaa 〜 +eee すべてをユーザに提示して欲しい場合は、‘mew-refile-ctrl-multi’ を ‘t’ に, +aaa だけを提示して欲しい場合は、‘nil’ に設定します。

また、+aaa 〜 +ddd は提示して欲しいけれどもそれ以降はいらない、つまり、+eee を提示するのは先に実行された関数群が何も推測できなかったときだけにしたい場合は、‘mew-refile-ctrl-multi’ を ‘t’ にして、‘mew-refile-guess-by-folder’ と ‘mew-refile-guess-by-default’ の間に ‘mew-refile-ctrl-throw’ を入れて下さい。

‘C-uo’ は、この推測の流れを Message バッファに表示します。

6.9 自動で整理整頓

毎日メッセージをたくさんもらう人は、まだ整頓していないメッセージを+inbox フォルダに大量に溜めてしまうことがあります。そんなときは、「メッセージ達よ、とにかく +inbox フォルダからどこかに行ってしまえ」と叫びたくなることがあるでしょう。Mew は、そんなわがままなを満足させるための自動整理整頓関数を提供しています。:) ‘M-o’ がその呪文です。

この関数を実行すると、現在のフォルダ内の特定のメッセージに対して、自動的に ‘o’ マークを付けてくれます。特定のメッセージとは、‘mew-refile-auto-refile-skip-any-mark’ が ‘nil’なら、‘o’ や ‘D’ マークが付いていないメッセージです。‘mew-refile-auto-refile-skip-any-mark’ が ‘t’ なら、なにもマークが付いてないメッセージです。‘mew-refile-auto-refile-skip-any-mark’ のデフォルト値は ‘nil’ です。また、‘C-u’ 付で呼び出すと、‘mew-refile-auto-refile-skip-any-mark’ の値とは関係なく ‘*’ マークの付いたメッセージを対象にします。

整頓先の決定には、先に説明した推測関数群が働くようになっています。この関数は、‘o’ を付けるだけですので、‘x’ を押さない限り実際にメッセージがどこかに行ってしまうことはありません。

Mew の整頓先推測はあまりにも賢すぎるので、この機能には仇となってしまいす。というのは、Mew が推測機能をフルに使って勝手に整頓してしまうと、大抵のユーザはどこにメッセージが整頓されたか分からなくなってしまうのからです。:) そのために、Mew が使う推測関数を制限する機能が提供されています。前に出てきた宣言をもう一度思い出して下さい。

```
(defvar mew-refile-guess-control
  '(mew-refile-guess-by-alist
    mew-refile-ctrl-throw
    mew-refile-guess-by-newsgroups
    mew-refile-guess-by-folder
    mew-refile-ctrl-throw
    mew-refile-ctrl-auto-boundary
    mew-refile-guess-by-thread
    mew-refile-ctrl-throw
    mew-refile-guess-by-from-folder
    mew-refile-ctrl-throw
    mew-refile-guess-by-from
    mew-refile-ctrl-throw
    mew-refile-guess-by-default))
```

‘mew-refile-guess-control’ の中に ‘mew-refile-ctrl-auto-boundary’ という関数があります。これがその仕掛です。自動整理整頓のときに限り、Mew はこの関数より下に記述してある推測を無視します。‘mew-refile-ctrl-auto-boundary’ より上に記述している関数が何も推測できなかった場合は、そのメッセージには ‘o’ が付きません。破滅が訪れる前に ‘mew-refile-ctrl-auto-boundary’ の御札を貼って下さい。

7 お目当てのメッセージを選択するには

たとえば、Subject: に party という文字が含まれているメッセージを見つけ出したり、From: が kazu@mew.org であるメッセージを選択したいと思うことがあります。このように、入力した条件にあるメッセージを見つけ出すコマンドを Mew は 3 つ提供しています。

‘?’ 入力した条件に合うメッセージに ‘*’ マークを付ける。現在の Summary モードの一覧を利用する。‘?’ を押した後に、条件を入力して下さい。

‘/’ 入力した条件に合うメッセージを Summary モードに一覧表示する。現在の Summary モードの一覧は上書きされます。‘/’ を押した後に、フォルダ名と条件を入力して下さい。

‘V’ 複数のフォルダから入力した条件に合うメッセージを見つけ出して Virtual モードに一覧表示する。操作は、See Section 7.2 [virtual], page 41 を参照して下さい。

以下条件の入力方法と Virtual モードに付いて解説します。

7.1 条件の入力方法

Mew が条件の入力をユーザに促すときは以下のように訊いてきます。

```
pick pattern:
```

以下に示すキーワードを組み合わせて条件を入力して下さい。

‘field=string’
フィールド field に文字列 string が含まれているときマッチ。フィールド field が、head、body、および、all の場合は、それぞれヘッダ全体、本文、メッセージ全体を意味する。

‘<pattern1> & <pattern2>’
<pattern1> かつ <pattern2> のときマッチ。

‘<pattern1> | <pattern2>’
<pattern1> または <pattern2> のときマッチ。

‘! <pattern>’
<pattern> でないときマッチ。

‘(<pattern>)’
<pattern> の内容を先に評価。

以下に例を示します。

(a) From: に kazu か含まれるメッセージ

```
from=kazu
```

(b) To: が mew、または、Cc: が mew であるメッセージ

```
to=mew | cc=mew
```

(c) To: が mew、または、Cc: が mew で、かつ、from が kazu であるメッセージ

```
(to=mew | cc=mew) & from=kazu
```

後は類推して下さい。

7.2 Virtual モード

Virtual モードは、複数のフォルダから入力した条件に合うメッセージを選びだし、単一の仮想的なフォルダとして扱います。Summary モードで 'V' を押すと、Virtual モードが作れます。

最初に、仮想フォルダ名を訊かれます。

```
Virtual folder name (virtual) :
```

任意の文字列を入力して下さい。単に 'RET' を押すと、"++virtual" になります。次に、単数または複数のフォルダ名を入力します。複数のフォルダを入力する場合は、"," で区切って下さい。もちろん、'TAB' で補完できます。

```
Folder name (+inbox) : +inbox, +mew
```

そして、条件を入力します。

```
pick pattern:
```

すると仮想フォルダができあがります。Virtual モードは、整頓や消去、検索など一部のコマンドを除いて、Summary モードと一緒です。仮想フォルダはまさしく仮想であり、ファイルシステムなどには存在しないことに注意して下さい。Emacs を終了すると、仮想フォルダはなくなります。

8 一休み

Mew を終了させたり、一時中断させたり、そのモード (正確にはバッファ) を消去する方法を以下に示します。

<Summary モードと Virtual モード>

‘q’ Mew を一時中断し、他のバッファに切り替えます。Mew 用のバッファはすべて残っていますから、バッファ操作で選べば再開できます。

‘Q’ Mew を終了します。Mew が使ったすべてのバッファを消去します。

‘C-cC-q’ そのモード (正確にはバッファ) を消去します。

<Draft モード>

‘C-cC-q’ その草稿を消去します。

<"*Mew watch*" バッファ>

‘C-cC-q’ そのバッファを消去します。

9 自分好みの Mew にするには

ここでは、Mew のデフォルトの動作を変更し、自分好みの Mew にする方法を説明します。主に "~/.emacs" で設定して下さい。

9.1 初級

ここでは、以下の変数について説明します。

- mew-draft-mode-hook
- mew-from
- mew-fcc
- mew-cc
- mew-dcc
- mew-window-use-full
- mew-summary-show-direction
- mew-summary-mark-direction

Draft モードでは、フックを 'text-mode-hook'、'mew-draft-mode-hook' の順で評価します。'text-mode-hook' で 'auto-fill-mode' を設定していない人は、'mew-draft-mode-hook' を以下のように設定するのがよいかもしれません。

```
(setq mew-draft-mode-hook (function (lambda () (auto-fill-mode 1))))
```

電子メールの管理者が設定しているのとは違うアドレスで送信したいことがあります。たとえば、管理者の設定が甘く、不要なホスト名が付いてしまう場合などです (この場合は管理者に頼み設定を修正してもらうのが一番です)。Mew では草稿のヘッダに From: アドレスがあれば、それがそのまま From: となります。補完を使いながら書いて下さい。いつも草稿に From: を用意するには、以下の例のように 'mew-from' を指定して下さい。

```
(setq mew-from "Kazu Yamamoto (山本和彦) <Kazu@Mew.org>")
```

From: 行を自分で指定できるということは、簡単にだれかになりすませるということです。くれぐれもこの機能を使ったいたずらをしないで下さい。また、他人も容易にだれかになりすませることに注意して下さい。大切な用件は、PGP/MIME で保護して送りましょう。

毎回 Fcc: でバックアップを取りたい人は、以下を .emacs に設定して下さい。

```
(setq mew-fcc "+Backup")
```

自分へ Cc:(Dcc:) したい人は 'mew-cc'('mew-dcc') を設定して下さい。

いつも Emacs のフレーム全体で Mew を使いたい人は、以下の設定をして下さい。

```
(setq mew-window-use-full t)
```

Summary モードの 'SPC' は、'mew-summary-show-direction' によって、次のメッセージの表示の仕方が変わります。また同様に、強いマークを付けた後カーソルが動く方向は 'mew-summary-mark-direction' により指定できます。以下の値を設定できます。

'up' 上のメッセージを表示する。

'down' 下のメッセージを表示する。

'next' 読み進めている方向の次のメッセージを表示する。

'stop' 次は表示しない。

両方ともデフォルトは ‘next’ です。いつもメッセージを下から読む人は、以下のようにすればよいでしょう。

```
(setq mew-summary-show-direction 'up)
```

9.2 中級

ここでは、以下の変数について説明します。

- mew-use-highlight-cursor-line
- mew-use-highlight-mouse-line
- mew-use-highlight-mark
- mew-use-highlight-header
- mew-use-highlight-body
- mew-use-highlight-url
- mew-use-highlight-x-face

‘mew-use-highlight-cursor-line’ が ‘t’ の場合、Summary モードでカーソルのある行に下線が引かれます。デフォルトは ‘t’ です。

XEmacs では ‘mew-use-highlight-mouse-line’ が ‘t’ の場合、Summary モードでマウスのある行に色がつきます。真中のボタンをクリックしながら、マウスだけでメッセージを読む際にはとても便利です。XEmacs でのデフォルトの値は、‘t’ になっています。

‘mew-use-highlight-mark’ が ‘t’ で、かつ、‘mew-highlight-mark-folder-list’ に列挙したフォルダが Summary モードである場合、マークの付いている行に色がつきます。‘mew-highlight-mark-folder-list’ のデフォルトは ’("+inbox") であり、‘mew-use-highlight-mark’ のデフォルトは ‘t’ です。‘mew-highlight-mark-folder-list’ を ‘t’ に設定すると、すべてのフォルダで色が付くようになります。

‘mew-use-highlight-header’ が ‘t’ の場合、Message モードと Draft モードでヘッダが色付けされます。デフォルトは ‘t’ です。

‘mew-use-highlight-body’ が ‘t’ の場合、Message モードと Draft モードで本文が色付けされます。デフォルトは ‘nil’ です。

‘mew-use-highlight-url’ が ‘t’ の場合、Message モードの URL を示す文字列が強調されます。デフォルトは ‘t’ です。

‘mew-use-highlight-x-face’ が ‘t’ で、かつ XEmacs を使っている場合、ヘッダ中の X-Face: が Message モードでアイコン化されます。XEmacs でのデフォルトは ‘t’ です。

9.3 上級

ここでは、以下の変数について説明します。

- mew-header-alist
- mew-cite-fields
- mew-cite-format
- mew-cite-prefix

いつも入れて欲しいヘッダは、‘mew-header-alist’ に連想リスト形式で定義して下さい。以下に例を示します。

```
(setq mew-header-alist
      '(("X-fingerprint:" . "6B 63 38 88 67 5E 96 8E  CE A4 62 73 3F 11 64 94")
        ("X-URL:" . "http://www.mew.org/~kazu/")))
```

引用ラベルは、引用するフィールドを ‘mew-cite-fields’、ラベルの書式を ‘mew-cite-format’ に定義します。引用記号は ‘mew-cite-prefix’ に指定します。デフォルトは、以下のようになっています。

```
(defvar mew-cite-fields '("From:" "Subject:" "Date:"))
(defvar mew-cite-format "From: %s\nSubject: %s\nDate: %s\n\n")
(defvar mew-cite-prefix "> ")
```

引用ラベルに Message-ID: を加え、ユーザ名付の引用記号にするには、以下のようにします。

```
(setq mew-cite-fields '("From:" "Subject:" "Date:" "Message-ID:"))
(setq mew-cite-format "From: %s\nSubject: %s\nDate: %s\nMessage-ID: %s\n\n")
(setq mew-cite-prefix-function 'mew-cite-prefix-username)
```

9.4 フック

Mew で用意されているフックをまとめます。

‘mew-env-hook’
: Mew の起動時の環境が設定される前に評価される。

‘mew-init-hook’
: Mew の起動時に評価される。

‘mew-summary-mode-hook’
: Summary モードに入るときに評価される。

‘mew-virtual-mode-hook’
: Virtual モードに入るときに評価される。

‘mew-message-mode-hook’
: Message モードに入るときに評価される。

‘mew-message-hook’
: メッセージが Message モードで表示される度に評価される。

‘mew-addrbook-mode-hook’
: アドレス帳の登録モードに入るときに評価される。

‘mew-draft-mode-hook’
: Draft モードに入るときに評価される。

‘mew-draft-mode-newdraft-hook’
: Draft モードにおいて新しい草稿が用意された際に評価される。

‘mew-draft-mode-reedit-hook’
: Draft モードにおいて古い草稿を再編集した際に評価される。

‘mew-cite-hook’
: Draft モードで引用する際に呼ばれる。通常 supercite を設定するために使う。

‘mew-before-cite-hook’
: Draft モードでメッセージを引用する直前に呼ばれる。

‘mew-make-message-hook’
: Draft モードで MIME メッセージを作る前、つまり ‘C-cC-m’ の際の最初に評価される。例：(add-hook ’mew-make-message-hook ’ispell-message)

‘mew-send-hook’
: メッセージを送信する前に評価される。現在では無意味であり、互換性のために残してある。

‘mew-real-send-hook’
: メッセージを送信する前に評価される。

‘mew-quit-hook’
: Mew の終了時に評価される。

‘mew-suspend-hook’
: Mew を一時中断した際に評価される。

‘mew-summary-inc-sentinel-hook’
: imget が終了する際に評価される。

‘mew-summary-scan-sentinel-hook’
: imls が終了する際に評価される。

‘mew-summary-exec-hook’
: Summary モードの ‘x’ の実行が終了する際に評価される。

‘mew-syntax-format-hook’
: マルチパートの書式を作成する関数 ‘mew-syntax-format’ が呼ばれた際に評価される。

9.5 Config

IM の "Config" ファイルに case 文を記述すると imget や imput の挙動を変化させられます。たとえば、Config が以下のように記述されていたとしましょう。

```
Imget.Src=pop/apop:kazu@mail.mew.org
User=kazu
FromDomain=Mew.org
case wide
Imget.Src=pop/rpop:robby@mx.wide.ad.jp
User=robby
FromDomain=wide.ad.jp
case iijlab
Imget.Src=imap:kazu@mailbox.iijlab.net
FromDomain=iijlab.net
```

通常では、imget は APOP を使って mail.mew.org から kazu のメッセージを読み込みます。また、imput は送信されるメッセージに From: がない場合 (mew-from が nil の場合など)、ユーザ名である kazu と Mew.org を @ で連結した文字列を From: に指定します。

もし –config=wide が指定されると、imget は RPOP を使って mx.wide.ad.jp から robby のメッセージを読み込みます。また、imput は From: に対し robby@wide.ad.jp を選ぶようになります。

このように imget や imput は、–config オプションで指定された文字列をもとに、有効な case 文を選択します。上の例から明らかですが、この機能を使うと読み込むメールボックスや送信時の From: を変更できて便利です。

Mew は起動時に Config の case 文を調べます。もし、1 種類以上の case がある場合、‘mew-config-list’ に設定します。–config が指定されない場合に読まれる部分は、default という文字列で表現されます。上の例では、‘mew-config-list’ には、’("default" "wide" "iijlab") が設定されます。

Summary モードで ‘C’ を押すと、‘mew-config-list’ を補完の候補として利用しながら、imget に渡す –config の値を選択できます。デフォルトでは "default" です。もし他の値が設定されていた場合、たとえば "wide" が設定されていた場合、‘i’ は次のように表示します。

```
Getting +inbox (wide)...
```

imput に case の選択を指示するには、送信するメッセージのヘッダ中の Config: フィールドで指定できます。Draft モードでは、See Section 4.1 [header], page 12 で説明したように、Config: というフィールド名を ‘TAB’ で補完できます。また、‘mew-config-list’ の値を ‘TAB’ で補完可能です。さらに、‘C-cTAB’ を使えば、‘mew-config-list’ の値を循環的に補完できます。

しかし、自分で値を選ぶのはめんどうです。Mew は整頓先のフォルダを推測するのだから、Config: の値も推測して欲しいと思うでしょう。もちろん、Mew は Config: の値を推測できます。Config: の推測ルールは、‘mew-config-guess-alist’ に設定できます。書式は、See Section 6.2 [by-alist], page 34 で説明した ‘mew-refile-guess-alist’ と同じです。

1 つ例を挙げてみましょう。

```
(setq mew-config-guess-alist
   '(("To:"
      ("wide.ad.jp" . "wide")
      ("mew.org" . "mew"))))
```

この場合、To: が wide.ad.jp に合致すれば "Config: wide" を、mew.org にマッチすれば "Config: mew" を挿入します。

この推測を利用して Config: を挿入できるタイミングは以下の 3 つです。

1. ‘mew-config-insert-when-prepared’ が ‘t’ なら、草稿を用意した時点。
2. ‘mew-config-insert-when-composed’ が ‘t’ なら、‘C-cC-m’ でメッセージを作成した時点。
3. ‘C-cC-o’ と入力した任意の時点。

参考までに書きますが、ホスト名で Config を変更したい場合は、以下の行を".emacs" に入れるとよいでしょう。

```
(setq mew-config-guess-alist
   (list (cons nil (system-name))))
```

10 アイコンのある生活

XEmacs で Mew を使えば、アイコン・ベースのインターフェイスでメッセージを読み書きできます。アイコン・ベースのインターフェイスは、従来のキー入力によるインターフェイスと親和性が高いように設計されています。

アイコン・インターフェイスの使い方は.....、説明するまでもありません。直感的にわかるはずです。でも少しだけ手ほどきを。

Summary、Virtual、および、Draft モードの標準のアイコンに束縛されている機能を利用するには、左ボタンをクリックして下さい。

マルチパートのメッセージを読み書きする際に表示されるアイコンは、左ボタンのクリックでそのバーを表示し、右ボタンを押すとポップアップ・メニューが表示されます。このメニューのおかげで、パートに対しさまざまな処理が可能となっています。

マルチパートのアイコンは、デフォルトで標準のアイコンの右に表示されます。左に表示するのが好きな人は、以下のように設定して下さい。

```
(setq mew-multipart-icon-position 'left)
```

11 メッセージの作法

メッセージをやりとりする際には、最低限のマナーがあります。マナーを守っていないメッセージは読みにくいため、大変損します。簡潔かつ適切な文章を書き、なるべく相手に理解してもらえるよう努力しましょう。

メッセージを書く際に気を付けるべき項目を以下に示します。

To: と Cc: を正しく書く
: To: が目的の人で、Cc: は参考までに送り付ける人です。自分のアドレスが Cc: の場合は、読み飛ばす人がいますから注意して下さい。また、不必要なアドレスを To: や Cc: に書いて、相手に迷惑をかけないようにしましょう。

To: と Cc: に書くアドレスの数は少なくする
: アドレスをたくさん To: や Cc: に書くのはよくないことです。面倒でもメーリングリストを作りましょう。

Subject: には本文の内容を的確に短く書く
: Subject: を見て読むか決める人がいますから、不適切な Subject: だと読んでもらえないかもしれません。長い Subject: は読みにくいので止めましょう。

改行は行末を、空行は改段落を表す。1 行は半角 70 文字程度にとどめる
: 意味もなく 1 行おきに書いたり、なん行にも渡って改行しないのは読みづらいです。とくに行が長いと引用するのに困ります。また、行頭にいくつかの空白文字を入れて右によせて書く人がいますが、これも無意味です。テキストの表示は、使っているコンピュータでまちまちですから、結局自分のコンピュータでみためがよくても、他人のコンピュータでそうであるとは限らないのです。

必要な部分だけを引用する
: 面倒でも不要な部分は削りましょう。Mew を使っていれば、引用は楽勝のはずです。

シグニチャは簡素にする
: 長いシグニチャは単なる自己満足です。

いたずらメッセージを送らない
: こんなことは注意したくありませんが、それでも不幸のメッセージなどを送ってくる人がいます。人格を疑われることを理解すべきです。

相手の読めるデータのみを添付する
: なんの合意もなしに送ってよいデータはテキストのみです。それ以外のデータを送る場合は、あらかじめ相手に送ってよいか確認をとりましょう。メーリングリストには、テキストのみを投稿するのが無難でしょう。

また、インターネットのマナーは RFC1855 を読むといいでしょう。See Chapter 22 [Bib], page 66 を参照して下さい。著者は、正確な文章の書き方について学生のためにまとめた入門書を WWW で公開しています。興味があれば、以下の URL にアクセスして下さい。

```
http://www.mew.org/~kazu/doc/japanese.html
```

12 MIME ってなぁに？

今までのメッセージ、正確には RFC822 メッセージは、本文にテキストしか格納できない規格でした。MIME は RFC822 を拡張した多目的メッセージです。

MIME は、ヘッダに

```
MIME-Version: 1.0
```

というフィールドを持ちます。このフィールドがない場合は、RFC822 メッセージです。MIME では、データの型を示す Content-Type: と符号化方式を示す Content-Transfer-Encoding: が重要なフィールドです。以下ではこれらのフィールドや MIME の特長について説明します。

12.1 データの型指定

MIME では、Content-Type:(以下 CT:) というフィールドにデータの型を指定できます。以下は、本文が US-ASCII である MIME の例です。

```
MIME-Version: 1.0
Content-Type: Text/Plain; charset=us-ascii
Content-Transfer-Encoding: 7bit
Subject: hello
From: Kazu

Hi all,
```

CT: が省略された場合は、Text/Plain; charset=us-ascii として取り扱われます。また、CT: Text/Plain のときに、charset が省略されると US-ASCII と解釈されます。

このように MIME では、CT: がテキストの場合は、charset で文字コードを指定できます。日本語には ISO-2022-JP を使います。

MIME では、本文に複数のデータを格納できます。これをマルチパートといいます。マルチパートのそれぞれのパートは、コンテントヘッダとコンテントボディから構成されています。CT: はヘッダだけでなく、コンテントヘッダ中にも現れます。逆に、ヘッダは特殊なコンテントヘッダだと考えても構いません。

詳しくは、See Section 12.3 [mime-multi], page 52 を参照して下さい。

以下に重要な CT: を示します。

‘`Text/Plain`’
: テキスト

‘`Message/Rfc822`’
: MIME を含むメッセージ。ヘッダと本文という構造がある。

‘`Multipart/Mixed`’
: マルチパート

‘`Application/Postscript`’
: PostScript

‘`Application/Octet-Stream`’
: バイトストリーム。バイナリファイルと思ってよい。

‘Image/Gif’
: GIF

‘Image/Jpeg’
: JPEG

‘Audio/Basic’
: AU 形式の音声ファイル

‘Video/Mpeg’
: MPEG

‘Message/External-Body’
: メッセージの外部に実体がある

12.2 安全な符号化

以前からバイナリを配送するために uuencode という符号化プログラムが使われていました。uuencode は、8 ビット 3 文字を 6 ビット 4 文字に変換しますが、変換後にたくさんの記号が現れます。これらの記号はメッセージのヘッダで特殊な意味を持つものが含まれており、ヘッダの拡張のためには利用できません。

また、空白文字も使われているのも厄介です。なぜなら、BITNET のファイルシステムには、行末に空白がありえないのです。もし、uuencode で符号化したときに、行末にたまたま空白が現れたとしましょう。これを BITNET のメッセージゲートウェイが受け取ると、当然行末の空白を削ってしまいます。よって、受信者は元のバイナリファイルを復元できません。

そこで、MIME では本文用に 2 つの符号化方式を定めました。

Base64 符号化方式
: "0-9A-Za-z/+" の 64 文字を用いて、8 ビット 3 文字を 6 ビット 4 文字に変換する。元々は PEM で考え出された。

Quoted-Printable 符号化方式
: 表示不可能な文字を "=" に続けて 16 進表記する。

各コンテントヘッダ中の Content-Transfer-Encoding:(CTE:) で符号化方式を指定します。取り得る値は以下の通りです。

7bit
: 無変換。7 ビットの行から構成される。

8bit
: 無変換。8 ビットの行から構成される。

binary
: 無変換。8 ビットのデータ・ストリーム。

base64
: Base64 で符号化した。7 ビットの行から構成される。

quoted-printable
: Quoted-Printable で符号化した。7 ビットの行から構成される。

CTE: が省略された場合は ‘7bit’ として扱われます。

ISO-2022-JP は 7 ビットの文字コードですから、CTE: は 7bit です。つまり、CTE: は省略して構いません。もちろん、base64 や quoted-printable で符号化しても構いませんが、フォルダにあるメッセージを more などで直接読めなくなるので、お勧めではありません。

12.3 マルチパート

CT: が Multipart である場合、そのコンテントボディには複数のデータが格納されることを意味します。データの境界は boundary に指定された文字列で区切られます。以下に例を示します。

```
Message-Id: <13060.789566615@mew.org>
From: Kazuhiko Yamamoto =?ISO-2022-JP?B?GyRCOzNLXE9CSScbKEI=?=
        <kazu@mew.org>
Subject: =?ISO-2022-JP?B?GyRCPC8kTjMoGyhC?=
To: m-sakura@ccs.mt.nec.co.jp
Mime-Version: 1.0
Content-Type: Multipart/Mixed; boundary=simple
Content-Transfer-Encoding: 7bit

--simple
Content-Type: Text/Plain; charset=iso-2022-jp
Content-Transfer-Encoding: 7bit

奈良名物「鹿」の絵を送ります。

--かず

--simple
Content-Type: Image/Gif
Content-Transfer-Encoding: base64
Content-Description: "Deer on the Nara park"

R0lGODdhFwG8ANUAABETDCoYDC8lFi4dJxcnKTMwLkUUC04uG2opEkgeJ04yMWg4Ly1FLVJG
NWdSMywyTks1Tmc3RjdRVjNcalRMUG9UU1xbY051eG9pcIcxEp5bM8d1NI1VSJhrVrRwUpR0
cKZ1dcN9WXuHOWmHc7WJN6yLbcyEWNCZdDZjjml0i5t7im+TmGeRonWly5aLlrCLlK+arJmn
pbettMabktWumM+zsrnCrtTLua21ycq6x6/J3NbQ1+bk29na5dzp8+7w8ywAAAAFwG8AAAG
/8CLcPhYtVgNyirWasZYEgDhIWGxRiXWcTIATHS/Hs6K2+1wt59azYtdJnBhKrVaWYcp7==

--simple--
```

この例では、"simple" という文字列で区切られています。boundary に指定された文字列には、先頭に "–" が付きます。最後の区切りには、後ろにも "–" が付きます。

各パートは、コンテントヘッダとコンテントボディから構成されます。両者は、ヘッダと本文のように空行で区切られます。逆にいうと、ヘッダと本文は、それぞれ特殊なコンテントヘッダとコンテントボディです。

テキスト以外を MIME で送信する場合は、必ずマルチパートを利用するようにしましょう。たとえば、本文にいきなり Audio/Basic を格納できますが、そんなメッセージを受け取ったらびっくりします。パート 1 に説明のテキスト、パート 2 に Audio/Basic を入れた方が親切でしょう。

マルチパートは、入れ子構造にできます。つまり、マルチパートのマルチパートなども作成できます。

ちなみに境界ですが、前後の改行まで含みます。上記の例では、"CRLF–simpleCRLF" が区切りです。

12.4 ヘッダの拡張

ヘッダはメッセージの配送に関わる情報を格納しているため、配送プログラムが誤動作するような文字列を入れるべきではありません。MIME では、フィールド値に ASCII 以外の文字列を格納する場合、以下のような形式で符号化し、安全な文字列に変えて挿入します。

```
=?<charset>?<encoding>?<encoded-string>?=
```

指定できる <charset> は CT: Text/Plain の charset と同じです。<encoding> には、'B' と 'Q' 符号化方式があり、前者は Base64 符号化方式、後者は Quoted-Printable 符号化方式の亜種を意味します。

ISO-2022-JP には、'B' 符号化方式が奨励されていますが、'Q' 符号化方式でも構いません。しかし、'Q' 符号化方式に対応しているインターフェイスはあまりないようです (もちろん Mew は対応しています)。

たとえば、Subject: に「山本和彦」と書いた場合、以下のように符号化されます。

```
Subject: =?ISO-2022-JP?B?GyRCOzNLXE9CSScbKEI=?=
```

上記の形式で符号化してもよいのはフィールド値であって、パラメータ値ではありません。パラメータ値の符号に使ってはならない理由としては、キーワードとなっている '=' が、パラメータ名とパラメータ値の区切り文字と重なっていることが挙げられます。パラメータを符号化するには、別の形式を利用します。以下に、「日本語のファイル」というファイル名をパラメータ値に指定した例を示します。

```
Content-Disposition: attachment;
 filename*=iso-2022-jp''%1B%24BF%7CK%5C8l%24N%25U%25%21%25%24%25k%1B%28B
```

13 嗚呼漢字コード

ここでは、非 ASCII 文字をメッセージで使うために人々が奮闘してきた歴史を漢字を例に取って振り返ります。

13.1 電子メールと地域化

1982 年、互換性を保証するために、電子メールの規格 RFC822 が記述されました。電子メールはアメリカ育ちであったため、残念ながら本文やヘッダには、ASCII 文字列しか格納できない規格でした。

しかし、英語以外の言語を母国語としている人達にはとても不便です。そこで、配送に関わるヘッダはともかく、本文に母国語を格納するため RFC822 はさまざまな国で拡張されました。

ヨーロッパ諸国では、ウムラウト (アクセント) 文字を表す 8 ビット 1 文字のコード Latin 1 がよく使われるようになりました。Latin 1 は ISO-8859-1 と呼ばれることがあります。

日本では、7 ビット 2 文字の JIS コード、UNIX でよく使われる 8 ビット 2 文字の EUC コード、パソコンで使われている 8 ビット 2 文字の SJIS コードが存在しました。日本のインターネットの前身である JUNET の先駆者達は、配送のためのコードとして JIS コードを ESC シーケンスで切り替える、いわゆる JUNET コードを選びました。

JUNET コードは ISO-2022-JP と呼ばれることがあります。ISO-2022-JP を使えば、複数の文字コードを切り替えるだけでなく、使われている文字コードが何かという情報を得られます。

Latin 1 や ISO-2022-JP に見られる本文の拡張は、あくまで地域に限定された紳士協定です。RFC822 を使う限り、地域間を越えてメッセージをやりとりするには、結局英語を使うしかないのです。

RFC822 は記述が曖昧なので、ヘッダや本文に 7 ビット文字である ISO-2022-JP を入れてもよいように読めます。たぶん、この説明を読めば誤解が解けるでしょう。「RFC822 は、ヘッダと本文のシンタックス (構文) を 7 ビット、それらのセマンティックス (意味) を US-ASCII と定めています。ISO-2022-JP のシンタックスは RFC822 に従っていますが、セマンティックスは RFC822 に違反しています。」

13.2 MIME の登場

絵や音声などを配送したい、地域化された RFC 822 の架け橋となる規格が欲しいなどの要望を満たすために、1992 年に MIME が規定されました。MIME では、テキストの文字コードを charset というパラメータに指定できます。たとえば、ISO-2022-JP は以下のように指定します。

```
Content-Type: Text/Plain; charset=iso-2022-jp
Content-Transfer-Encoding: 7bit

日本語のテキスト
```

この charset は役に立つのでしょうか？ もちろんです。charset は、インターフェイスに正しくテキストの文字コードを伝える役割を果たします。ノルウェーの人が ISO-8859-1 で日本にメッセージを送ってきたとしましょう。ISO-8859-1 に対応しているインターフェイスな

ら、対応するフォントで表示すればいいし、対応していないなら無視すればいいのです。Mew では、Mule の内部コードに変換する関数の引数に charset を利用しています。

「ISO-2022-JP の上位互換規格であり、さまざまな文字コードを格納できる ISO-2022-JP-2 などを使えば charset など要らない。なぜなら ISO-2022-JP-2 自体に文字コードの情報が含まれているから」という人がいます。しかし、これらの人は MIME を理解できているとは思えません。

確かに理想的にいえば、この主張は正しいのです。しかし、MIME は現実主義です。MIME はユーザが明日から ISO-2022-JP-2 を使うようになるという大胆な仮定はしていません。また、世の中に存在する脆弱な配送プログラムでも安定して動くように考慮されています。世の中は、そんなにお行儀のよいプログラムばかりではないし、たくさんの資源を使えるほど豊かでもないのです。「明日から UNICODE を使え」と言われたらいやでしょう？

MIME は、さまざまに地域化されたメッセージの架け橋として charset を用意しました。MIME で増えた作業は、charset の挿入だけであり、今まで通り我々は ISO-2022-JP を使えます。ISO-2022-JP-2 を標準にしたいなら、ISO-2022-JP-2 をデフォルトで使う地域を広げていけばよいのです。この意味で、ISO-2022-JP-2 と MIME は相反してはなく、逆に、ISO-2022-JP-2 の普及のために MIME を利用できると言えます。もちろん文字コードの符号化方式である ISO-2022-xx に charset という単語が相応しくないのは、MIME の開発者も認めているところです。

MIME を使えば、非 ASCII 文字を ASCII 文字列に符号化し、ヘッダに挿入できます。このような枠組で、電子メールの配送プログラムの誤動作を防止し、また、ヘッダに英語以外の言語を書くことを実現しているのです。もう、Subject: に「日本語を書いてはいけません」なんて言わなくてよくなりました。:)

MIME は地域化された RFC 822 を禁止する規格ではありません。よって、MIME のインターフェイスは、以下のような動作が望まれています。

読むとき

1. ユーザがデフォルトの charset を選べるようにしておく。
2. MIME-Version: がないメッセージの場合は、本文をデフォルトの charset として扱う。
3. MIME-Version: があり、Content-Type: がない場合は、US-ASCII として扱う。
4. MIME-Version: と Content-Type: がある場合は、Content-Type: に指示された charset を利用する。

書くとき

1. MIME-Version: と Content-Type: Text/Plain の charset を必ず付ける。
2. charset には、最小限の文字集合を選ぶようにする。たとえば、英語だけなら US-ASCII を選ぶようにする。これを守らないと、読めるべきメッセージが読めなくなる可能性がある。たとえば、US-ASCII だけなのに、ISO-2022-JP と書いてあると、US-ASCII にしか対応してないメーラで読めないかもしれない。

スプールやフォルダに ISO-2022-JP を EUC-Japan に変更して格納する場合は、無条件に変換してはいけません。きちんと charset を確かめ、ISO-2022-JP だけを EUC-Japan に変換するようにして下さい。

ヘッダに非 ASCII 文字を挿入する機能は MIME の一機能ですが、実際には、MIME-Version: フィールドは必要ありません。

13.3 正規化の概念

残念ながら、世の中のコンピュータは、データをさまざまな方法で表現します。以下によく利用されている OS とその行末を示します。

- UNIX :: LF(0x0a)
- MS-DOS :: CRLF(0x0d0a)
- MacOS :: CR(0x0d)

よって、行末に関して取り決めがないと、これらの OS 間では安全にテキストが交換できません。RFC822 では、メッセージの再送の際に行末を CRLF に変換することになっています。このように、共通の書式への変換を「正規化」といいます。SJIS や EUC-Japan を ISO-2022-JP に直すのも正規化の一種です。

さて、PGP の暗号化や署名ついて考えてみましょう。たとえば、Mac のユーザが行末が CR である文章に署名し UNIX ユーザに送ったとします。UNIX ユーザが行末を LF に変換し署名を確認したとしたら、検証が失敗するのは明らかでしょう。そこで、PGP への入力はあらかじめ正規化されている必要があるのがお分かりになると思います。

PGP で暗号化したり署名したりする場合は、まずテキストを ISO-2022-JP に変換し、行末を CRLF に直して下さい。

14 Mew のこだわり

Mew の精神を一言で表すと、

> Mew wants something simple or nothing at all.

となります。つまり、複雑な機能は要らないのです。単純で吟味された機能しか提供しません。

たくさんの方が「昔使っていたメーラにはこんな機能があって便利だから、Mew にも付け加えて欲しい」といいます。しかし、こんな説明では著者を説得するのは難しいでしょう。なぜなら、Mew は他のプログラムが今までやっていないことを実現しようとしているからです。「昔はこうだった」という理由はあまりにも力不足です。

もし自分の意見が正しいと思うのでしたら、根気強く著者を説得して下さい。著者は忙しいので昔説明されたことはすぐ忘れてしまいますし、人間ですから思い込みや偏見があります。素晴らしいアイディアを理解できないことが多分にあるでしょう。どうか、くじけないで下さい。

15 Mew の来た道

役には立ちませんが、Mew の歴史をひもといてみましょう。

15.1 mh-e からの脱却

1993 年の秋に WIDE Project の FJPEM の実験に参加しました。そのころ、櫻井さんが FJPEM を mh-e で利用しやすくするため、mhpem を作りました。mhpem に触発され、1994 年の冬は修論の逃避行動を兼ねて mhpem の改良に励みました。

mhpem は暗号メッセージを自動的に復号化するおしゃれなプログラムでしたが、いくつか問題がありました。大きな問題は、mh-e が簡単に拡張できない柔軟性に欠けたプログラムだったことです。復号化したメッセージは、次に読むときは速くなるよう保存したいと思いますが、mh-e を改良するのは難しいのです。mh-e がバージョンアップし、mhpem が新しい mh-e で動かなかったとき、著者は切れました。

復号化した PEM を保存したい。MIME も保存したい。どうして、簡単に MIME を取り扱えるインターフェイスがないのか。なぜ、複数のメッセージから簡単に引用できないのか。マークを駆使して面白いことができないなんて。メッセージの整頓が大変なのはいやだ。結局、mh-e を改良することでは、自分の欲しいプログラムは書けなかったのです。

mhpem の自動復号機能、mhasync の非同期な scan 機能、ウインドウの大きさを動的に変更できる GNUS の機能、マークを付けて複数のニュースを取り扱える gnus-mark の機能、VM のような整頓機能、解析したメッセージの保存、美しく柔軟性のあるプログラミングスタイル.... さまざま断片が著者の中で 1 つにまとまり始めました。1994 年 3 月、奈良への引越し前のことでした。

15.2 Mew の誕生

1994 年 4 月、本格的に Mew の製作に取り掛かりました。一覧表示の終了を待たなくてよい機能、動的なウインドウ設定、マーク、解析したメッセージの保存などは、お手本があったので早くから実装されていました。

はじめの頃は MIME を MIME モードで表示していました。マルチパートのメッセージで 'SPC' を押すと、Summary モードから MIME モードに移動していたのです。しかし、歌代さんは言いました、「なぜ MIME モードかあるの？ Summary だけで十分ではないか」。目から鱗の落ちる思いでした。

このころの大きな障害は、メッセージの整頓方法と MIME の作成方法でした。

確かに VM のように、ユーザにいちいち Lisp を書かせれば、整頓は簡単になります。しかし、ユーザにこまめに設定させること自体が嫌だったのです。「メッセージのヘッダから無限の可能性のある整頓先を模索するのは馬鹿だ。限りのある実際のフォルダの中から候補を選べばいい」。このアイディアを思いついたときは、次の朝が来るのが待ち遠しかったものです。整頓機能は、後に乃村さんによって強化されました。

複雑な MIME を簡単に作成するにはどうすればよいか？ ユーザには作成文法とか MIME の書式などの理解を押しつけるのはあんまりだ。一言説明すれば、だれでも直感的に MIME を作れるようにしたい。この答えをくれたのは門林さんです。「MIME の構造ってファイルシステムに似ているよね」。そうです、シングルパートをファイル、マルチパートをディレクトリと考えればよいのです。ファイル操作はだれにでもできるし、ファイル構造を MIME に変換する仕事は Mew が請け負います。

15.3 PGP との出会い

FJPEM の実装に疑問を感じていた著者は、1994 年の初夏 PGP と出会いました。そのころは PGP 2.5、つまり、MIT が RSAREF を使って、RSA の特許に抵触しない非商用の PGP を模索していた時期です。PGP 2.6 のリリースによって、Phil と RSADSI は和解しましたが、RSAREF を使っている 2.6 は米国国外への持ち出しが禁止されています。

しかし、2.3a を基にした 2.6ui の存在を知り、PGP ユーザになりました。PGP を始めて使ったときの感動は忘れられません。本当に計算し尽くされたプログラムです。1995 年の冬、サンディエゴで開かれた通称 ISOC Security Symposium(NDSS) という国際会議に参加しました。このとき、プロシーディングと共に O'Reilly から発売されてすぐの Simson の PGP という本をもらいました。久々に読みふけった傑作中の傑作です。このような理由から、PEM と MIME の統合よりも、PGP と MIME の統合に力を注ぐようになりました。PGP/MIME は現在実験段階であり、標準化にはまだ時間がかかります。

「暗号化とか署名はマークで表せばいい」というアイディアは、櫻井さんと情報処理全国大会の論文を書いたときに教えてもらいました。マークならいつでも好きなときにキャンセルできるという利点を理解するのは少し時間がかかりましたけど。

15.4 MH からの独立

「うまい寿司を食わせてやるからみんな福岡に集合だ」のかけ声のもと、1997 年 4 月にそうそうたるプログラマが福岡ドームのある百道に結集しました。九州システム情報技術研究所にハック部屋を借り、近くのハイヤット・リージェンシーを寝床にして、3 泊 4 日に及ぶ MH からの独立作戦が開始されました。

とにかく食べる、気に入るまでハックする、とことんまで議論するという作業をなんども繰り返し、福岡を後にするころにはほぼ Mew が MH から独立し、IM を使うようになっていました。(この週のエンゲル係数が高かったのは言うまでもありません。)

その後もメーリングリストで議論を戦わせ、合意を取り、またハックするという作業を繰り返し、最後に歌代先生が高速化の魔法を唱えて、ようやく 7 月頭にベータリリースにこぎつけました。

15.5 ニュースの統合

1994 年秋の WIDE 合宿で Mew の BOF(Birds Of a Feather::井戸端会議) をしました。この BOF で佐野さんは「電子メールとニュースの中間がメーリングリストだよね」という忘れられない言葉を残しました。闇の中で手探り状態だったところに、一筋の光を見た思いです。Perl 5 が安定してきた現在、ニュースの統合は間近に迫っています。

16 Mew の行く道

Mew 1.95 がバージョン 1 としては最後になる予定です。以下に Mew 1.95 で実装する機能を挙げます。

真の多言語対応
: Mew の多言語対応は今でも素晴らしいのですが、8bit のメッセージをうまく転送できない、韓国語の取り扱いが難しいなどの欠点があります。これらを欠点を修正して真の多言語対応を目指します。

MIME エディタ
: 現在、マルチパート・メッセージを再編集するのは困難です。Mew 1.95 では、あたかも新規メールを扱っているかのような再編集機能を実現します。

カスタマイズ
: ‘`defcustom`’ などを利用して、カスタマイズを容易にします。

Draft モードの定型書式とサブモード
: Draft モードで定型の文書をあらかじめ用意したり、携帯電話メールへメッセージを書くために便利なモードを用意したりします。

Mew 2 では IM から独立する予定です。このプロジェクトは 「Mew 2 の逆襲」と呼ばれています。

IMAP デーモン
: Mew 1.9x の欠点の 1 つは、IM を何度も呼び出すので、Perl が起動し IM のファイル群を読む時間その都度待たされることです。そこで、ローカルのコンピュータに常駐する IMAP デーモンを作成し、Mew は IMAP デーモンと通信するようにします。

全文検索のデータベース
: 全文検索のためのデータベースに対応します。実は現在でも Namazu が利用できます。"contrib/00readme-namazu.jis" を参照して下さい。

親子関係のデータベース
: メッセージの親子関係を保存するデータベースに対応します。これによってスレッドが実現できます。また、メッセージの未読管理もできるようになります。ひょっとすると、メッセージの整頓という作業はまったくなくなるかもしれません。

ネットニュース
: そろそろネットニュースにも対応しないといけませんね。

17 入手方法とメーリングリスト

ここでは、Mew の入手方法とメーリングリストについて触れます。

17.1 Mew の入手方法

Mew の最新バージョンは以下から入手できます。

`ftp://ftp.Mew.org/pub/Mew/mew-current.tar.gz`

ときどきサンプルメッセージが以下のように提供されます。

`ftp://ftp.Mew.org/pub/Mew/samples.tar.gz`

17.2 メーリングリスト

新しいバージョンは、

`mew-release@Mew.org`

で英語でアナウンスしています。入りたい人は

`mew-release-ctl@Mew.org`

宛に本文に "#help" と書いて電子メールを送って下さい。mew-release には著者以外が投稿できないように制限が加えられています。

Mew の質問や議論などは、

`mew-dist@Mew.org`

に日本語で投稿できます。mew-dist へ入りたい人は

`mew-dist-ctl@Mew.org`

宛に本文に "#help" と書いて電子メールを送って下さい。mew-dist は mew-release に含まれているので、mew-dist に登録すれば、自動的に mew-release 宛の電子メールを受け取るようになります。

Mew の質問はできるだけ mew-dist へお願いします。著者には受け取ったすべての質問に答えている時間はありません。mew-dist へ質問すると、他の人が答えてくれることを期待できます。

18 謝辞

まずはじめに、櫻井三子さんにお礼を述べたいと思います。彼女が mhpem を作り触発してくれなかったら Mew を作ることはなかったからです。(もうなくなってしまいましたが) mew-url や mew-pem の多くの部分を書いています。また、複雑なメッセージの作成にマークを使うというヒントも与えてくれました。

Summary モードをもっと簡素にと助言して頂いた歌代和正さん、MIME とファイル構造って似てるよねと言ってくれた門林雄基さん、いつも電子メールに関して貴重な助言をしてくれる中村素典さん、電子メールとニュースを統合するきっかけを与えてくれた佐野晋さんに心から感謝します。

いつも素晴らしいコードを提供しくれる酒井清隆さん、メッセージの整頓機能を賢くしてくれた乃村能成さん、こまめに英語を直してくれる牛島幹友さん、info を書く気にさせてくれた島慶一さん、ありがとうございます。

I would like to thank David Worenklein for contributing many codes. I'm grateful to Atsushi Shionozaki, Darren Stalder, and David Worenklein for proofreading my paper. I'd like to acknowledge to Scandinavia guys for their good suggestions.

IM の作成に協力してくれた萩野純一郎さん、笠原義晃さん、中村素典さん、乃村能成さん、西和則さん、太田英憲さん、IM を高速にしてくれた歌代和正さんに感謝します。Mew を OS/2 で動かす努力とたくさんのテストをしてくれる奥西藤和さんにお礼を申し上げます。Mew が Win95/WNT で動くようになったのは、主に北口修一さんと山口修平さんの努力によるところが大きいです。Mew が XEmacs で楽しいのは、寺西裕一さんのおかげです。

Mew に対し献身的な白井秀行さん、鯉江英隆さん、そして後藤俊一さんに感謝します。

Mew はたくさんの人の貢献の上に成り立っています。ここに名前を挙げていない方にも心から感謝しています。著者はなにぶん多忙ですので、すべての電子メールに答える時間が取れません。返事が帰って来なかった人は、本当にすみませんでした。

19 著作権について

Mew は以下の著作権に従います。

変更の有無にかかわらず、ソースおよびバイナリ形式の再配布および利用は、以下の条件を満たしていれば、これを許可する。

1. ソース・コードの再配布は、上記の著作権表示、この条件項目、および、以下の免責事項を保存しなければならない。
2. バイナリ形式の再配布は、上記の著作権表示、この条件項目、および、以下の免責事項を、その配布に付随する説明書、あるいはその他の資料のいずれかに明記しなければならない。
3. 前もって特別に許諾を得ない限り、このソフトウェアから派生した製品の推奨や販売促進のために、このチーム名と貢献者達の名前を利用してはならない。

このソフトウェアは「このままの形で」提供され、明示的あるいは言外の保証は、商用利用および特定目的への適合に対する言外の保証も含み、またこれらだけに限らず、存在しない。たとえ以下のような損害の可能性を示唆されていたとしても、どのような形にしろこのソフトウェアの利用から発生した問題において、このチームと貢献者達は、(代替製品やサービスの調達; 利用権、データ、あるいは利益の損失; あるいは営業の中断を含む、またこれらだけに限らず) 直接的に、間接的に、偶然に、特別に、懲罰上、あるいは、必然的に生じてしまった損害に対し責任はなく、いかなる責任理論上でも契約の有無に係わらず厳密な責任はなく、また (過失あるいはその他を含む) 不法行為に対しても責任はない。

CD ROM に入れて配布したいという方は、できるだけ教えて下さい。教えてくれなくても怒りませんが、教えて頂けるとありがたいです。配布を断ったことはありません。

この Info の著作権は、著者に属します。配布、利用共に自由ですが、無保証です。また、この Info から発生した被害に対し、著者は一切責任を負いません。

20 著者紹介

Kazuhiko YAMAMOTO // Kazu

山本和彦 // かず (くん)

1970 年、南を虹の松原を湛える海岸、東をウグイが泳ぐ川、北と西をなだらかな山に囲まれる山口県光市に生まれる。中学生のときに映画「ウォーゲーム」を見てセキュリティに関心を持つ。自然と遊びながら、高校まで光市で過ごす。

1988 年、九州大学工学部電気系入学のため福岡へ。都会の殺伐さ、汚れた海、言葉の違いに苦しみながらも卒業まで至る。3 年生のときに初めて電子メールに触れ、4 年生のころにインターネットに惹かれていく。Sun に入社渡米を試みるが、不幸にも米国のビザ発行ポリシーが厳しくなった時期と重なり、失敗。

1992 年、九州大学大学院工学研究科情報工学専攻へ進学。本格的にインターネットの研究に従事。経路制御を専門とする。セキュリティの普及活動を開始。修士 2 年の春に、書く書くと言ってなかなか書かなかった「ハッピーネットワーキング」をリリース。

1994 年、奈良先端科学技術大学院大学に助手として就任。約 4 年の大学教官生活中に、Mew の開発に取り組む。「若いうちにプログラムを書こう、教育は歳をとってからもできるよね」という結論に至り、1998 年 IIJ 技術研究所に新たな研究環境を求めた。

研究分野 :: メッセージ・システム、IP version 6
ひとこと :: 「うまく説明できないものは、本質的に優れていない」
ふたこと :: 「優しいものが優れている」

Email: Kazu@Mew.org
URL: http://www.mew.org/~kazu/
PGP fingerprint: 6B 63 38 88 67 5E 96 8E CE A4 62 73 3F 11 64 94

21 用語集

Mew で使う単語をまとめます。

‘フォルダ’ 受け取ったメッセージを保存するファイルやディレクトリ

‘Summary モード’
メッセージの一覧を表示するモード

‘Message モード’
テキストメッセージの内容を表示するモード

‘Draft モード’
メッセージを書いたり、作成したりするモード

‘MIME’ 本文にテキスト以外のオブジェクトを格納したり、ヘッダに非 ASCII 文字を挿入したりできる規格。MIME を使えば、本文にテキスト、絵、音声などを同時に取り込め、また、Subject: に日本語を書ける。Multipurpose Internet Mail Extensions の略称で、「多目的メール」と呼ぶことがある。詳しくは、See Chapter 12 [MIME], page 50 を参照。

‘PGP’ Phil Zimmermann 氏が作成した暗号メッセージや電子署名を実現するプログラム。Pretty Good Privacy の略。

‘ニュース’ ネットニュース、あるいは、USENET ニュースのこと。

‘メッセージ’
電子メールやニュース、あるいは、MIME などの総称。インターネットメッセージの省略語。

22 参考文献

ここでは、参考文献を紹介します。RFC(Request For Comments) は、インターネットで知識を共有するために書かれた文献です。たとえば、以下のサイトから入手できます。

`ftp://ftp.isi.edu/in-notes/`

インターネットでのマナーは、以下の RFC を読んで学んで下さい。

- S. Hambridge, "Netiquette Guidelines", RFC 1855, 1995

昔のテキスト・メールの書式や配送の仕組みは以下の RFC を参照しましょう。

- D. Crocker, "Standard for the format of ARPA Internet text messages", RFC 822, 1982
- J. Postel, "Simple Mail Transfer Protocol", RFC 821, 1982

MIME については、以下の RFC を読んで下さい。

- N. Freed and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, 1996.
- N. Freed and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types" RFC 2046, 1996.
- K. Moore, "MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text", RFC 2047, 1996.
- N. Freed, J. Klensin and J. Postel, "Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures", RFC 2048, 1996.
- N. Freed and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Five: Conformance Criteria and Examples", RFC 2049, 1996.
- R. Troost, S. Dorner, K. Moore, "Communicating Presentation Information in Internet Messages: The Content-Disposition Header Field", RFC 2183, 1997.
- N. Free and K. Moore, "MIME Parameter Value and Encoded Word Extensions: Character Sets, Languages, and Continuations", RFC 2231, 1997.

ニュースの書式と配送プロトコルは、以下の RFC で定義されています。

- M. Horton and R. Adams, "Standard for interchange of USENET messages", RFC 1036, 1987.
- B. Kantor and P. Lapsley, "Network News Transfer Protocol: A Proposed Standard for the Stream-Based Transmission of News", RFC 977, 1986.

PGP を学ぶなら、以下の書籍がいいでしょう。

- PGP: Petty Good Privacy, Simson Garfinkel, O'Reilly & Associates, Inc, 1995. (訳書:: "PGP 暗号メールと電子署名", Simson Garfinkel 著, 山本和彦監訳, 株式会社オライリージャパン, 1996.)

PGP と MIME の統合については、以下を参照して下さい。

- J. Galvin, S. Murphy, S. Crocker and N. Freed, "Security Multiparts for MIME: Multipart/Signed and Multipart/Encrypted", RFC 1847, 1995.
- M. Elkins, "MIME Security with Pretty Good Privacy (PGP)", RFC 2015, 1996.

メールの基礎知識は、Mew ニュースレターにまとめられています。http://www.Mew.org/ を参照して下さい。

23 変数索引

M

Short Contents

Table of Contents